

Vorbereitung – Vor dem Unterricht

<https://github.com/finalangel/modular-frontend>

Bereitet euch vor und ladet das folgende Repository herunter:

```
git clone https://github.com/finalangel/modular-frontend
```

Dient als Grundlage für den Unterricht, mitmachen lohnt sich!

– Bonus –

Installiert die Dependencies für **ex1**, **ex2** und **ex3** bereits um Zeit zu sparen:

```
cd ex1 && npm install  
cd ex2 && npm install  
cd ex3 && npm install
```

Komplexe Sites Modular Strukturieren

CAS Front End Engineering



Hi,
I am Angelo,
Nice to teach you!

Organisatorisches Sprache

```
.foo {  
  Höhe: 300px;  
  Außenabstand-unten: 10px;  
  Schriftgröße: 20px !wichtig;  
  Hintergrund-farbe: schwarz;  
  Farbe: weiß;  
}
```

```
.foo {  
  height: 300px;  
  margin-bottom: 10px;  
  font-size: 20px !important;  
  background-color: black;  
  color: white;  
}
```

Organisatorisches Agenda

- Einführung
- Grundlagen
- Frameworks
- Methodologien
- Beispiele

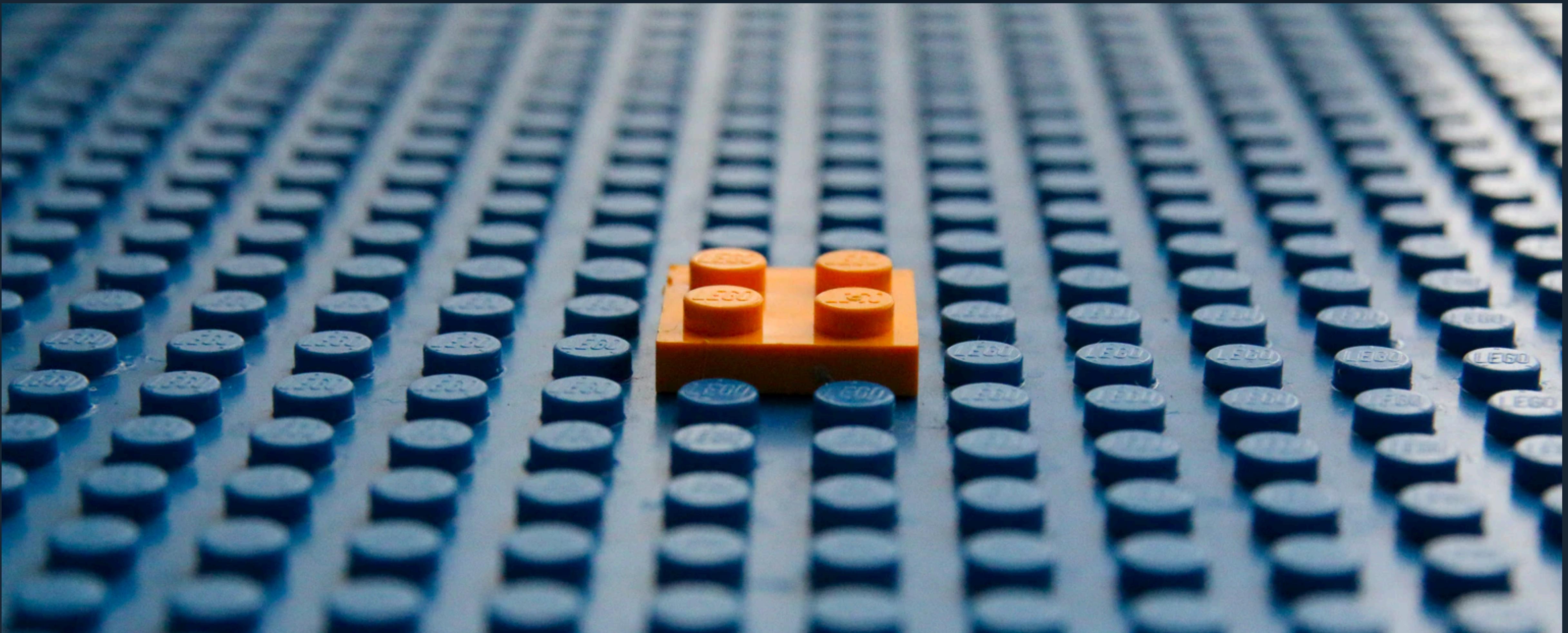
Organisatorisches

Lernziele

- Ihr kennt die fundamentale Theorie, wie Modulare Frontends mithilfe von HTML, CSS und JavaScript aufgebaut werden können.
- Ihr wisst, wie "Front-end Frameworks" und "Styleguides" eingesetzt werden und könnt euren eigenen Styleguide mithilfe von Storybook erstellen.
- Atomic Design, BEM, OCCS, SMACSS und ITSG sind keine Fremdwörter mehr für euch. Ihr könnt weiter in die Materie eintauchen und euren eigenen Stil finden.
- Ihr versteht die zahlreichen Beispiele und könnt eigenständig unterschiedliche Methodologien anwenden, sowie den vorgegebenen Code entsprechend überarbeiten.

Fragen bei Unklarheiten zu jeder Zeit stellen !

Die Modulare Front-end Entwicklung



Beispiel mit Lego

Analogie

Front-end

Teams

Depth

Performance

Die Module einer Webseite
verhalten sich ähnlich wie Legos.

Beispiel mit Lego

Analogie

Front-end

Teams

Depth

Performance



Analogie

Front-end

Teams

Depth

Performance

Beispiel mit Lego



Bezug zum Front-end

Analogie

Front-end

Teams

Depth

Performance

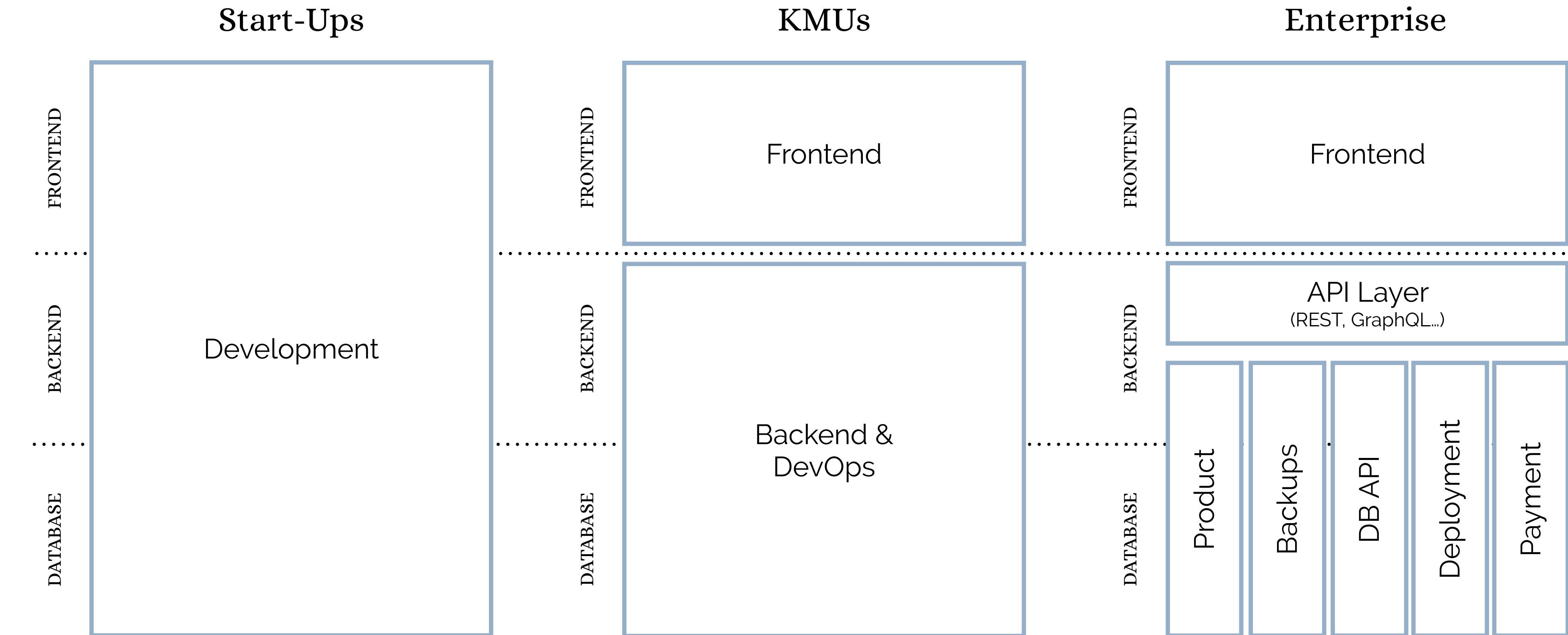
- Unser Code entwickelt sich über einen gewissen Zeitraum,
- und wird daher auch schnell inkonsistent, je nach Teamgrösse.
- Deshalb wählen wir schnell ein "Refactoring" als potentielle Lösung.
- Dadurch entsteht ein neuer Code / Version!

There are only two hard things in Computer Science:
Cache invalidation and naming things

– Phil Karlton

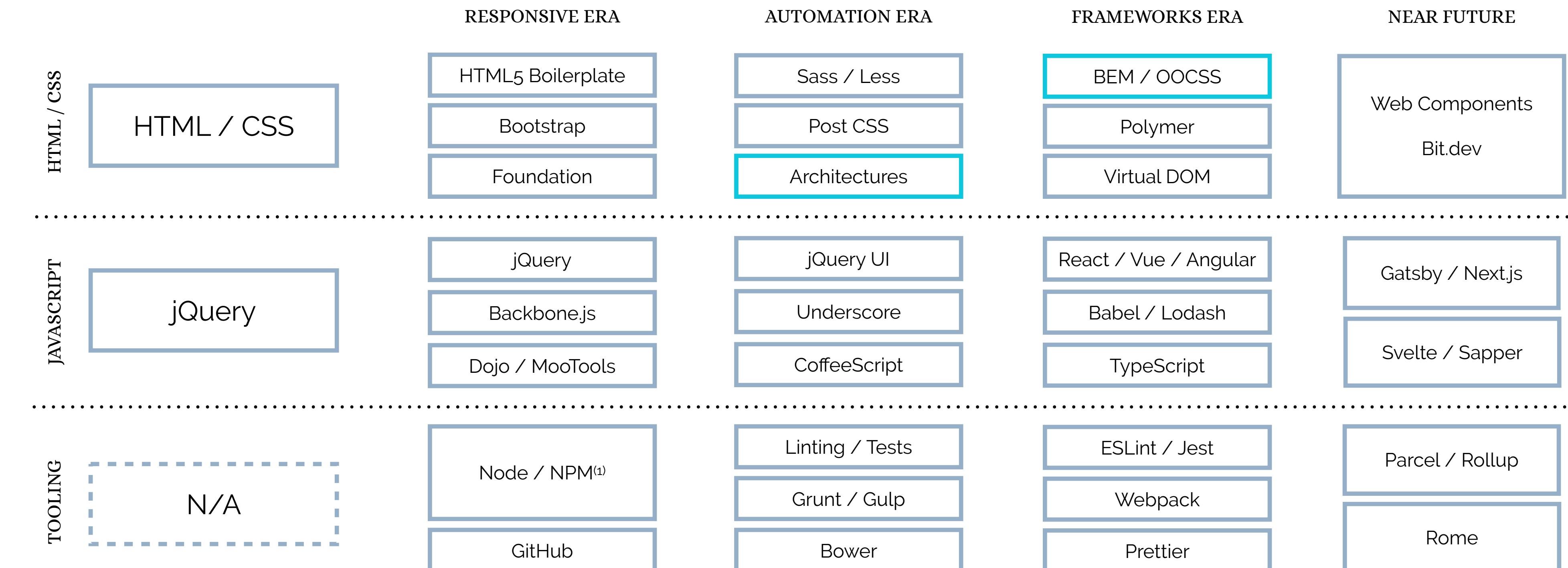
Teamentwicklung

Analogie
Front-end
Teams
Depth
Performance



Analogie
Front-end
Teams
Depth
Performance

Technical Depth



⁽¹⁾ Node wurde 2009 veröffentlicht, feierte seinen Durchbruch aber erst später.

Der ewige Kampf

Analogie

Front-end

Teams

Depth

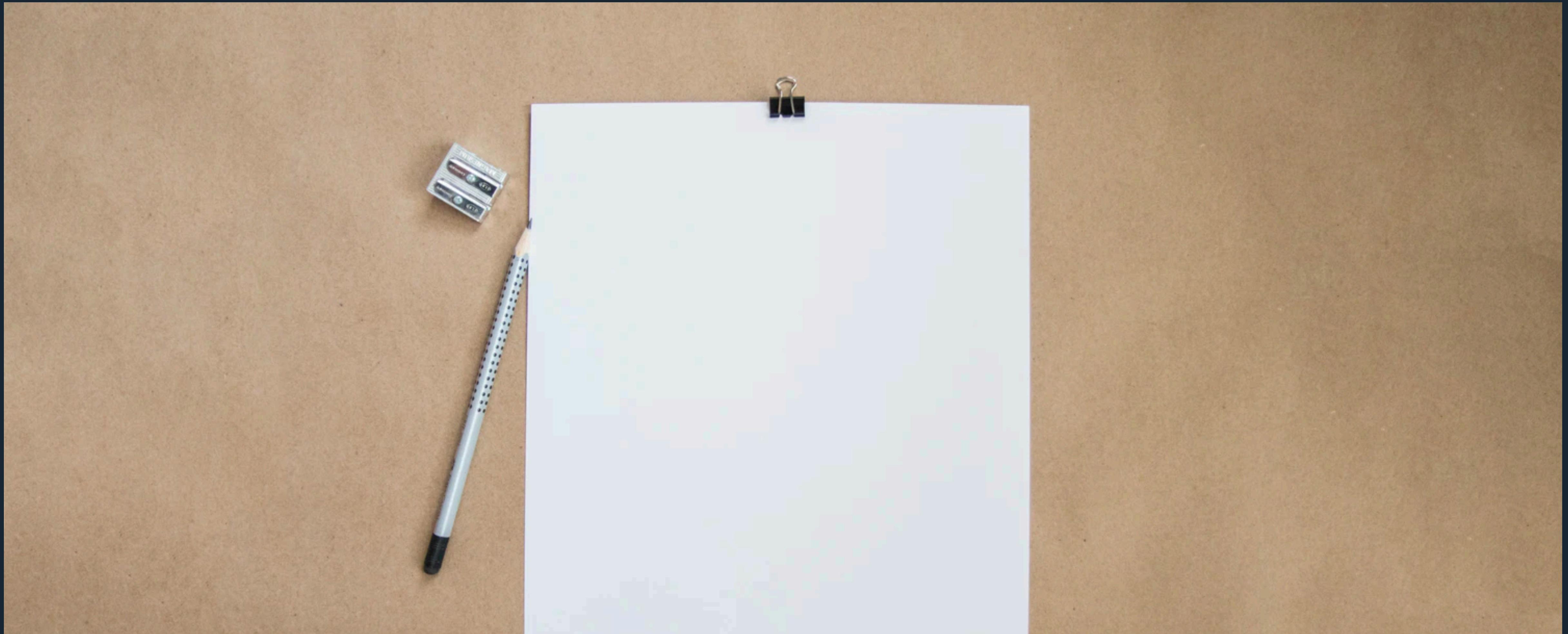
Performance

Performance

VS.

Modularität

Begleitübung



Begleitübung

<https://github.com/finalangel/modular-frontend>

- Grundlage für die weiteren Übungen
- `git clone` & mitmachen
- Übungen bauen aufeinander auf
- Parcel, React & Storybook
- Feedback bei Fehlern 😅

The screenshot displays a website template with the following structure:

- Header:** A navigation bar with links: Home, Event, Accessoires, Apparel, and Feautred Items.
- Main Content Area:** A large gray placeholder area labeled "1024 × 300".
- Section:** A title "A Big Title" followed by a sub-section "Intro Copy Headline" with placeholder text: "Fusce vel dui. In enim justo, rhoncus ut, imperdiet a, venenatis vitae, justo. Pellentesque commodo eros a enim. Fusce a quam. Etiam vitae tortor."
- Item Grid:** A row of four items, each consisting of a placeholder image (300 × 300), a title "Item Title", a category "Item Category", and a price "\$88.00".

HTML & CSS



HTML & CSS

HTML

Toolbox

Syntax

Konflikte

Templates

OO-Stil

SMA-Stil

Fazit

Tools

- caniuse.com

- validator.w3.org

- javascript.info/dom-navigation

CSS

CSS3, Sass und Post CSS

Layout resets & Frameworks

Animation Frameworks

CSS in JS

Tools

- sassmeister.com

- jigsaw.w3.org/css-validator

- framer.com/motion

HTML & CSS

HTML Syntax

Toolbox

Syntax

Konflikte

Templates

OO-Stil

SMA-Stil

Fazit

```
<!-- DOCTYPE HTML 4.01 Transitional -->  
  
<h1>This is my website</h1>  
  
<div id="content">  
  <h2>Getting my legos together</h2>  
  
  <div class="article">  
    <h3>Happy tree friends</h3>  
  </div>  
  <div class="article">  
    <h3>Happy tree friends</h3>  
  </div>  
</div>
```

```
<!-- DOCTYPE HTML -->  
  
<h1>This is my website</h1>  
  
<section>  
  <h2>Getting my legos together</h2>  
  
  <article>  
    <h3>Happy tree friends</h3>  
  </article>  
  <article>  
    <h3>Happy tree friends</h3>  
  </article>  
</section>
```

HTML & CSS

HTML Syntax

Toolbox

```
// JSX

import React from 'react';

const Content = () => (
  <>
    <h1>This is my website</h1>

    <div id="content">
      <h2>Getting my legos together</h2>

      <div className="article">
        <h3>Happy tree friends</h3>
      </div>
      <div className="article">
        <h3>Happy tree friends</h3>
      </div>
    </div>
  </>
);

export default Content;
```

Syntax

Konflikte

Templates

OO-Stil

SMA-Stil

Fazit

```
// Pug

doctype html
html(lang="en")
  head
    title=pageTitle
  body

    h1 This is my website

    .content
      h2 Getting my legos together

      if articles
        .article
          h3 Happy tree friends

        .article
          h3 Happy tree friends
```

HTML & CSS

CSS Syntax

Toolbox

Syntax

Konflikte

Templates

OO-Stil

SMA-Stil

Fazit

```
/* Sass */

@mixin reset { padding: 0; margin: 0; }
.heading { color: #f00; }
.content {
    h2, h3 {
        @extend .heading;
        font-size: 26px;
        line-height: 1.6;
    }
    .article {
        @include reset;
        h3 {
            font-size: 18px;
            line-height: 1.3;
        }
        &:hover {
            background: rgba(0, 0, 0, 0.2);
        }
    }
}

/* Compiled */

.heading, .sidebar h2, .sidebar h3 {
    color: #f00;
}
.content h2, .content h3 {
    font-size: 26px;
    line-height: 1.6;
}
.content .article {
    padding: 0;
    margin: 0;
}
.content .article h3 {
    font-size: 18px;
    line-height: 1.3;
}
.content .article:hover {
    background: rgba(0, 0, 0, 0.2);
}
```

HTML & CSS

CSS Syntax

Toolbox

```
// styled components                                // inline styles

import styled from 'styled-components';
const H1 = styled.h1`                           const h1 = {
    font-size: 26px;                            fontSize: '26px',
`;                                              `};

const CONTENT = styled.section`                const content = {
    display: flex;                            display: 'flex',
    flexDirection: row;                      flexDirection: 'row',
    background: #eee;                        background: '#eee',
`;                                              `};

const Container = ({ title, children }) => (
    <>
        <H1>{title}</H1>
        <CONTENT>{children}</CONTENT>
    </>
);
export default Container;
```

Syntax

Konflikte

Templates

OO-Stil

SMA-Stil

Fazit

HTML & CSS

HTML Probleme

- Browser Support (Polyfills)
- Section & Structure handling (Outline)
`<body>, <article>, <aside>, <footer>, <header>, <nav>, <section>`

Toolbox

Syntax

Konflikte

Templates

```
// HTML
```

```
<h1>This is my website</h1>  
<section>  
  <h2>Getting my legos together</h2>
```

```
// Generated outline
```

```
This is my website  
└ Getting my legos together  
    └ Happy tree friends  
    └ Happy tree friends
```

OO-Stil

```
<article>  
  <h3>Happy tree friends</h3>  
</article>  
<article>  
  <h3>Happy tree friends</h3>  
</article>  
</section>
```

SMA-Stil

Fazit

HTML & CSS

CSS Probleme

- Specificity

type selectors → class selectors → ID selectors → !important

- shame.css

Toolbox

Syntax

Konflikte

Templates

OO-Stil

SMA-Stil

Fazit

```
/* don't: !important */  
.box { font-size: 18px !important; }
```

```
/* don't: descendant combinator */  
div.box { background: white; }
```

```
/* don't: type selector */  
h2 { color: green; }
```

```
/* don't: descendant selector */  
.box h2 { color: black; }
```

```
/* don't: IDs for styles */  
#box { border: 1px solid black; }
```

```
/* bad */  
#content-container a { ... }
```

```
/* bit better */  
#content-container > a { ... }
```

```
/* kinda okay */  
.content-container .special { ... }
```

```
/* you're doing it right */  
.content-container-special { ... }
```

HTML & CSS

Weitere Probleme

Toolbox

Syntax

Konflikte

Templates

OO-Stil

SMA-Stil

Fazit

- Nutzt generische Namen: "box", "pagination", "panel"
- Mehrzahl für Kollektionen: "customers", "authors", "books"
- Beachtet Anwendungen: <a> vs <button> wird oft vergessen
- Nutzt Variablen und Dateistrukturen: "_colors.sass", "layout.mustache"
- Flache Hierarchie: Nesting ist toll wird aber schnell unübersichtlich
- 2020: line-height & select's still suck
- IDs sollten nur für 2 Zwecke genutzt werden:

HTML & CSS

Templates

Toolbox

```
// Django Templates

{% extends "base.html" %}

{% block title %}Welcome{% endblock %}

{% block "content" %}
  <p class="text-small">
    Lorem ipsum dolor sit amet, consectetur
    elit, sed do eiusmod tempor incididunt.
  </p>

  <section class="features">
    {% for entry in entries %}
      {{ entry }}
    {% endfor %}

  </section>
{% endblock "content" %}
```

Syntax

```
// EJS

<%- include('layout/title', {
  title: "Headline"
}); %>

<p class="text-small">
  Lorem ipsum dolor sit amet, consectetur
  elit, sed do eiusmod tempor incididunt.
</p>

<section class="features">
  <% entries.forEach(function(entry){ %>
    <%- include('feature/item', {
      entry: entry
    }); %>
  <% } ); %>
</section>
```

Konflikte

Templates

OO-Stil

SMA-Stil

Fazit

HTML & CSS

Object Oriented CSS

Toolbox

Syntax

Konflikte

Templates

OO-Stil

SMA-Stil

Fazit

Trennung von Struktur & Stil

- Box-Layout von Farben und Form trennen
- Gleiche Deklarationen zusammenfassen

Trennung von Container & Inhalt

- Duplikate vermeiden
- Gemeinsamkeiten ermitteln

HTML & CSS

Object Oriented CSS

Toolbox

```
/* chaos */

button {
    width: 200px;
    height: 50px;
    border: solid 1px #ccc;
    background: linear-gradient(#ccc, #222);
    box-shadow: rgba(0, 0, 0, .5) 2px 2px 5px;
}
```

Syntax

```
#box {
    width: 400px;
    overflow: hidden;
    border: solid 1px #ccc;
    background: linear-gradient(#ccc, #222);
    box-shadow: rgba(0, 0, 0, .5) 2px 2px 5px;
}
```

Konflikte

Templates

OO-Stil

SMA-Stil

Fazit

```
/* order */

.button {
    width: 200px;
    height: 50px;
}

.box {
    width: 400px;
    overflow: hidden;
}

.highlight {
    border: solid 1px #ccc;
    background: linear-gradient(#ccc, #222);
    box-shadow: rgba(0, 0, 0, .5) 2px 2px 5px;
}

.widget {
    @include .highlight;
}
```

```
#widget {
    border: solid 1px #ccc;
    background: linear-gradient(#ccc, #222);
    box-shadow: rgba(0, 0, 0, .5) 2px 2px 5px;
}
```

HTML & CSS

Scalable and Modular Architecture

Toolbox

Syntax

Konflikte

Templates

OO-Stil

SMA-Stil

Fazit

Unterteilung in:

Base ➔ Layout ➔ Module ➔ State ➔ Theme ➔ Changes

- Base

Generische/Globale Deklarierungen und Resets

- Layout

Globale Elemente wie Header, Footer und Sidebar, die mehrheitlich einmal vorkommen

- Module

Wiederverwendbare Elemente die mehrfach vorkommen

HTML & CSS

Scalable and Modular Architecture

Toolbox

Syntax

Konflikte

Templates

OO-Stil

SMA-Stil

Fazit

Unterteilung in:

Base → Layout → Module → **State** → Theme → Changes

- **State**

Modifikationen am Layout oder den Modulen

- **Theme**

Designanpassungen am Layout oder den Modulen

- **Changes**

Änderungen die durch dynamische Prozesse vorgenommen werden

HTML & CSS

Scalable and Modular Architecture

Toolbox

```
/* base */
body {
    color: black;
    font-size: 12px;
    text-decoration: none;
}

/* layout */
.content {
    display: flex;
    align-items: center;
    justify-content: center;
}

/* module */
.button {
    border: 1px solid black;
    &:hover {
        color: white;
        background: black;
    }
}

/* state */
.button-disabled {
    opacity: 0.5;
    &:hover {
        color: black;
        background: transparent;
    }
}

/* theme */
.button-marketing {
    color: red;
    border: 5px solid #green;
}

/* changes */
btn.click(() => btn.toggle('button-hidden'));

.button-hidden {
    display: none;
}
```

Syntax

Konflikte

Templates

OO-Stil

SMA-Stil

Fazit

HTML & CSS

Zusammenfassung

Toolbox

Syntax

Konflikte

Templates

OO-Stil

SMA-Stil

Fazit

Do One Thing and Do It Well

– Unix Philosophy

HTML & CSS – Begleitübung

<https://github.com/finalangel/modular-frontend>

- `cd ex1 && npm install && npm run start`
- Öffnet: <http://localhost:8000>
- `cd ex1 && npm run test` (neue Session)

Aufgabe

- Verbessert den Aufbau & Style nach eurem Geschmack
- Design darf nicht verändert werden (test-case)
- Dauer ca. 15-20 Minuten

The screenshot shows a modular frontend application. At the top is a header with navigation links: Home, Event, Accessoires, Apparel, and Feautred Items. Below the header is a large, light gray rectangular area containing the text "A Big Title" and "1024 × 300". Underneath this is a section titled "Intro Copy Headline" with placeholder text: "Fusce vel dui. In enim justo, rhoncus ut, imperdiet a, venenatis vitae, justo. Pellentesque commodo eros a enim. Fusce a quam. Etiam vitae tortor." Below this are four items, each consisting of a gray square placeholder (300 × 300) and a row of text: "Item Title", "Item Category", and "\$88.00".

JavaScript

JS

JavaScript

JS, JSX, TS, TSX

Toolbox

Syntax

Konflikte

Styling

Demeter

Fazit

- ECMAScript ➡ TypeScript; Webpack, Babel
- CommonJS, AMD (RequireJS), **ES6 Modules**
- **Frontend frameworks:** React, Angular, Vue.js, Svelte
- **Backend frameworks:** Express, Next.js, Gatsby
- **Data layers:** GraphQL, Apollo, Redux

Tools

- <https://www.npmjs.com/>
- <https://stateofjs.com/>

Toolbox

Syntax

Konflikte

Styling

Demeter

Fazit

```
var Calculate = function (numberArray) {
  this.numbers = numberArray;
};

Calculate.prototype = {
  sum: function () {
    return this.numbers.reduce(function (a, b) {
      return a + b;
    });
  },
  pow: function () {
    return this.numbers.reduce(function (a, b) {
      return Math.pow(a, b);
    });
  },
};

console.log(
  new Calculate([15, 15]).sum(),
  new Calculate([2, 2, 2]).pow()
)
/* results
   30, 16
*/
... without DOM ;)
```

JavaScript

Vanilla Syntax

JavaScript

jQuery Syntax

Toolbox

Syntax

Konflikte

Styling

Demeter

Fazit

```
<div className="box">
  <button
    class="btn"
    data-values="[15, 15]">
    Sum
  </button>

  <button
    class="btn"
    data-values="[2, 2, 2]">
    Pow
  </button>
</div>

... let's do this

/* results
   30, 16
*/
```

```
$('.btn')
.eq(0)
.click(function (event) {
  var sum = $(this)
    .data('values')
    .reduce(function (a, b) {
      return a + b;
    });
  console.log(sum);
});

$('.btn')
.eq(1)
.click(function (event) {
  var pow = $(this)
    .data('values')
    .reduce(function (a, b) {
      return Math.pow(a, b);
    });
  console.log(pow);
});
```

Toolbox

Syntax

Konflikte

Styling

Demeter

Fazit

JavaScript

React Syntax

```
import React from 'react';
import { sum, pow } from './utils';

const Calculate = () => {
  return (
    <div className="box">
      <button
        className="btn"
        onClick={sum([15, 15])}>
        Sum
      </button>
      <button
        className="btn"
        onClick={pow([2, 2, 2])}>
        Pow
      </button>
    </div>
  );
};

export default Calculate;
```

```
// utils.js

export function sum(numbers) {
  return numbers.reduce(function (a, b) {
    return a + b;
  });
}

export function pow(numbers) {
  return numbers.reduce(function (a, b) {
    return Math.pow(a, b);
  });
}
```

JavaScript

Probleme mit der Struktur

- Trennt Dynamische Funktionen vom Stil
- Vermeidet IDs als Referenzpunkt

Toolbox

Syntax

Konflikte

Styling

Demeter

Fazit

```
/* avoid */  
document.getElementById('box');  
  
/* preferred */  
document.querySelector('.box');  
  
/* ideal */  
document.querySelector('.js-action');  
  
/* dynamic function */  
btn.click(() => box.cls('box-hidden'));
```

```
/* styles */  
.box {  
    width: 200px;  
    height: 100px;  
}  
  
.box-hidden {  
    display: none;  
}  
  
.js-box {  
    /* no styles! */  
}
```

```
<!-- HTML -->  
<div class="box js-action">  
    Lorem ipsum dolor sit amet  
    <button class="btn js-close">  
        Close  
    </button>  
</div>
```

JavaScript

Weitere Probleme

Toolbox

Syntax

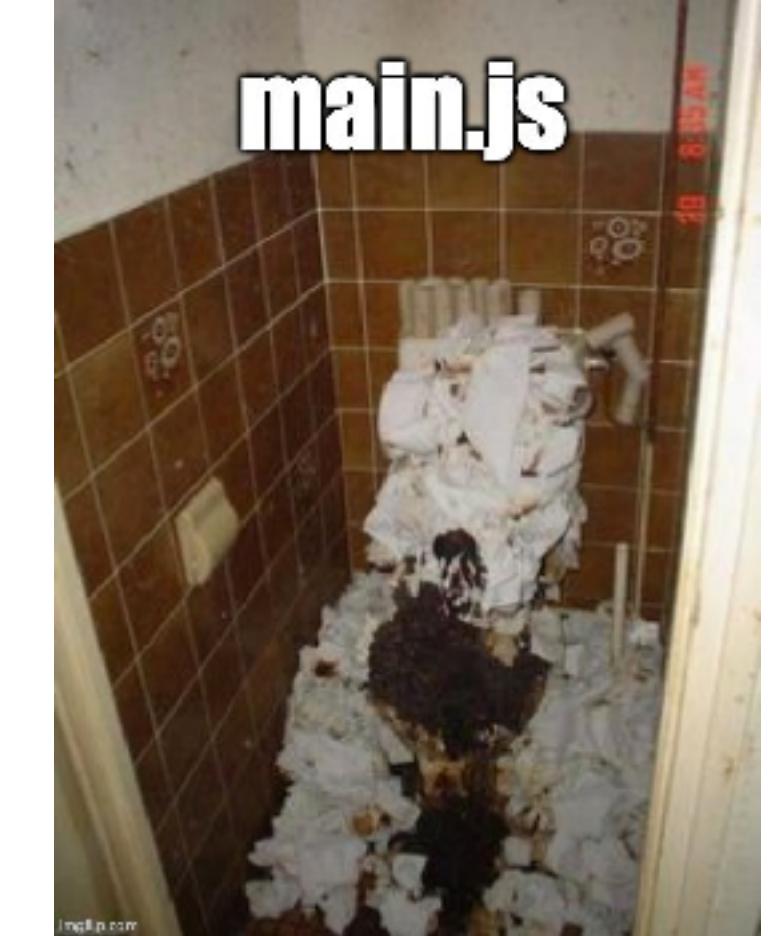
Konflikte

Styling

Demeter

Fazit

- Dateiaufteilung, nicht alles sollte in eine Datei/Klasse (**main.js**)
- Der Code-Umfang wird leicht zu Spaghetti Code
- Aber übertreibt es nicht, jeder Request zählt!
- Es gibt so viele Code Styles und Vorlieben...
- Fear of missing out - welche Frameworks / Libraries
- Wie suche ich widgets aus (date-picker, tooltips etc.)



JavaScript

Styled Components

Toolbox

Syntax

Konflikte

Styling

Demeter

Fazit

Funktionsweise

- Erstellt ein `<div>` und fügt eine willkürliche Klasse ein: "f7Hg3"
- CSS wird mittels inline styles im `<head>` oder als CSS Fragment geladen
- Achtung performance, wird zur "run time" ausgeführt
- Keine Konflikte mit anderen Styles
- UI Libraries können nicht benutzt werden (sharing)

```
// styled components

import React from "react";
import styled from "styled-components";

const StyledDiv = styled.div`  
  display: flex;  
  flex-direction: row;  
  background: #eee;  
`;  
  
const Container = ({ children }) => (  
  <StyledDiv>{children}</StyledDiv>  
);  
  
export default Container;
```

JavaScript

CSS Modules

Toolbox

Syntax

Konflikte

Styling

Demeter

Fazit

Funktionsweise

- Fügt eine teils willkürliche Klasse ein:
 "_container_33c5c"
- CSS wird als CSS Fragment geladen
- Achtung performance, wird zur "run time" ausgeführt
- Kann schlecht mit anderen Methoden kombiniert werden
- Keine Konflikte mit anderen Styles
- Erlaubt nur camelCase als Klassen-Namen

```
// css modules

import React from "react";
import styles from "./container.module.css";

const Container = ({ children }) => (
  <div className={styles.container}>
    {children}
  </div>
);

export default Container;

// > cat container.module.css
.container {
  display: flex;
  flex-direction: row;
  background: #eee;
}
```

JavaScript

Inline Styling

Toolbox

Syntax

Konflikte

Styling

Demeter

Fazit

Funktionsweise

- Wie inline Styling bei CSS
- borderRadius statt border-radius
- Bei komplexeren Styles (Media Queries etc.) wird es schwierig
- Kann ideal für Animationen (Framer Motion etc.) genutzt werden

```
// inline styles

import React from "react";

const styles = {
  display: "flex",
  flexDirection: "row",
  background: "#eee",
}

const Container = ({ children }) => (
  <div styles={styles}>
    {children}
  </div>
);

export default Container;
```

JavaScript

CSS & SCSS

Toolbox

Syntax

Konflikte

Styling

Demeter

Fazit

Funktionsweise

- Caching und gute Performance wird in Fragmente aufgeteilt
- Viele CSS Frameworks und grössere Bekanntheit
- Einfache Modulare Anwendung
- Kann schnell unübersichtlich werden
- Konfliktpotential mit anderen Komponenten

```
// inline styles  
  
import React from "react";  
  
import "./styles.css";  
import "./styles.scss";  
  
const Container = ({ children }) => (  
  <div className="container">  
    {children}  
  </div>  
);  
  
export default Container;
```

JavaScript

Das Gesetz von Demeter

Toolbox

```
// implementation

import React from "react";

import Button from "components/button";

const Container = ({ children }) => (
  <div className="container">
    <Button type="primary">
      Call to action
    </Button>
  </div>
);

export default Container;
```

Syntax

Konflikte

Styling

Demeter

Fazit

```
// components/button.js

import React from "react";

import { Button as BootstrapButton } from "react-bootstrap";

const Button = ({
  type,
  children,
  ...rest
}) => (
  <BootstrapButton variant={type} {...rest}>
    {children}
  </BootstrapButton>
);

export default Button;
```

JavaScript

Zusammenfassung

Toolbox

Syntax

Konflikte

Styling

Demeter

Fazit

- Klare Trennung von Stil und Dynamischen Funktionen
- Entwickelt unabhängige, wiederverwendbare Komponenten
- Erfindet das Rad nicht neu; Prettier ➔ Code Style, Webpack ➔ Bundling, NPM ➔ Widgets
- JavaScript entwickelt sich weiter, nehmt nicht immer das neuste aber spielt damit
- Styled Components **vs.** CSS Modules **vs.** Inline Styling **vs.** CSS Files

Simple is better than complex.

– Zen of Python

JavaScript – Begleitübung

<https://github.com/finalangel/modular-frontend>

- `cd ex2 && npm install && npm run start`
- Öffnet: <http://localhost:8000>
- `cd ex2 && npm run test` (neue Session)

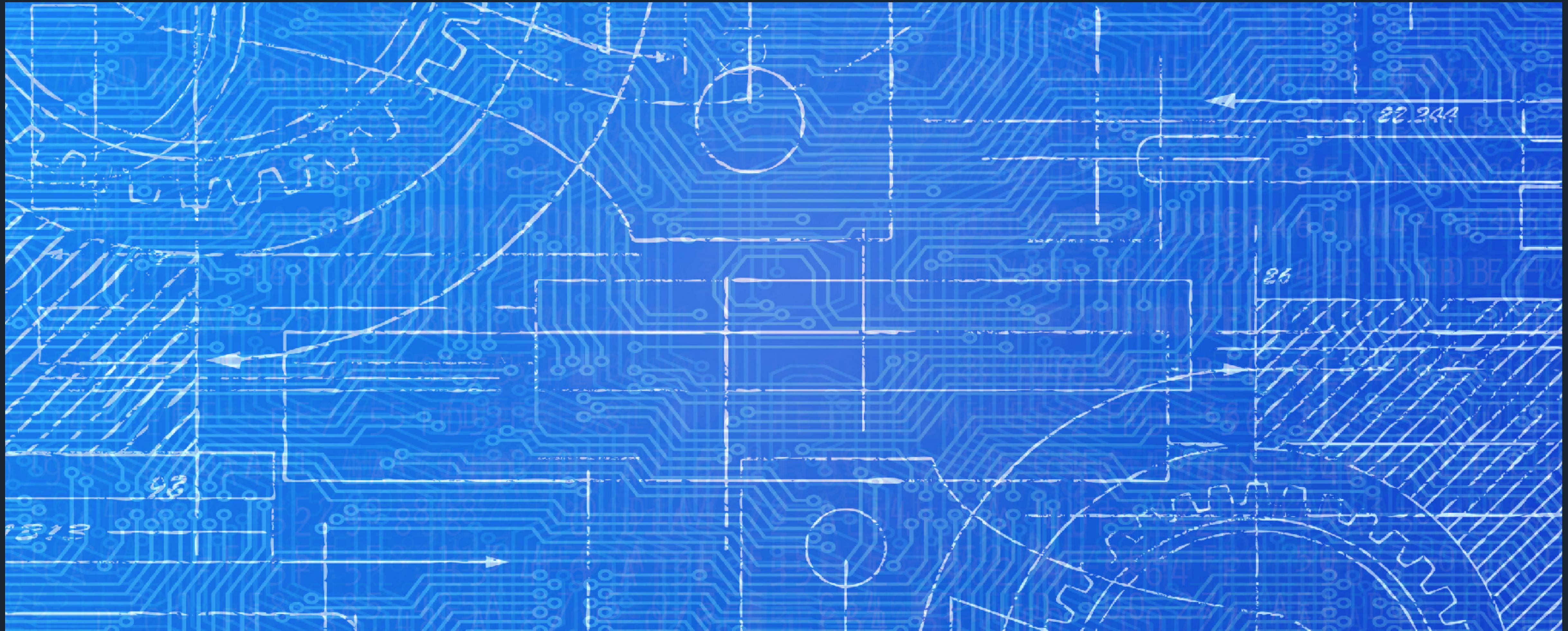
Aufgabe

- Verbessert den Aufbau & Style nach eurem Geschmack
- Design darf nicht verändert werden (test-case)
- Dauer ca. **15-20 Minuten**

The screenshot shows a modular frontend application. At the top is a header with a navigation bar containing "Home", "Event", "Accessoires", "Apparel", and "Feautred Items". Below the header is a large, light gray rectangular area labeled "1024 × 300". Underneath this is a section titled "Intro Copy Headline" with placeholder text: "Fusce vel dui. In enim justo, rhoncus ut, imperdiet a, venenatis vitae, justo. Pellentesque commodo eros a enim. Fusce a quam. Etiam vitae tortor." Below this are four items, each consisting of a small image placeholder (300 × 300), a title ("Item Title"), a category ("Item Category"), and a price ("\$88.00").

Item Title	Item Category	\$88.00
Item Title	Item Category	\$88.00
Item Title	Item Category	\$88.00
Item Title	Item Category	\$88.00

Grundbausteine für euer Front-end



Grundbausteine

Boilerplates vs. Framework vs. Library

Definition

Boilerplates

Frameworks/
Libraries

Styleguides

- Boilerplates

Geben die Struktur am Anfang des Projektes vor

- Framework⁽¹⁾

Gibt vor wie eine Applikation aufgebaut werden soll
(React, Angular, Django)

- Library⁽¹⁾

Funktionen, die einen Task ausführen

⁽¹⁾ Die Grenze zwischen Framework und Library ist oft verschwommen

Boilerplate oder nicht?

Pro

- Das Rad nicht neu erfinden
- Gute Dokumentation & Community
- Einstieg wird vorgegeben
- Schnelles Prototyping
- Best practice

Con

- Wo soll man beginnen
- Wie weiss ich wo was hin muss
- Viele Meinungen und Guidelines
- Unerwünschte Last
- Kann schnell unübersichtlich werden

Beispiele:

HTML5 Boilerplate, Create React App, Angular CLI, Parcel

Definition

Boilerplates

Frameworks/
Libraries

Styleguides

Grundbausteine

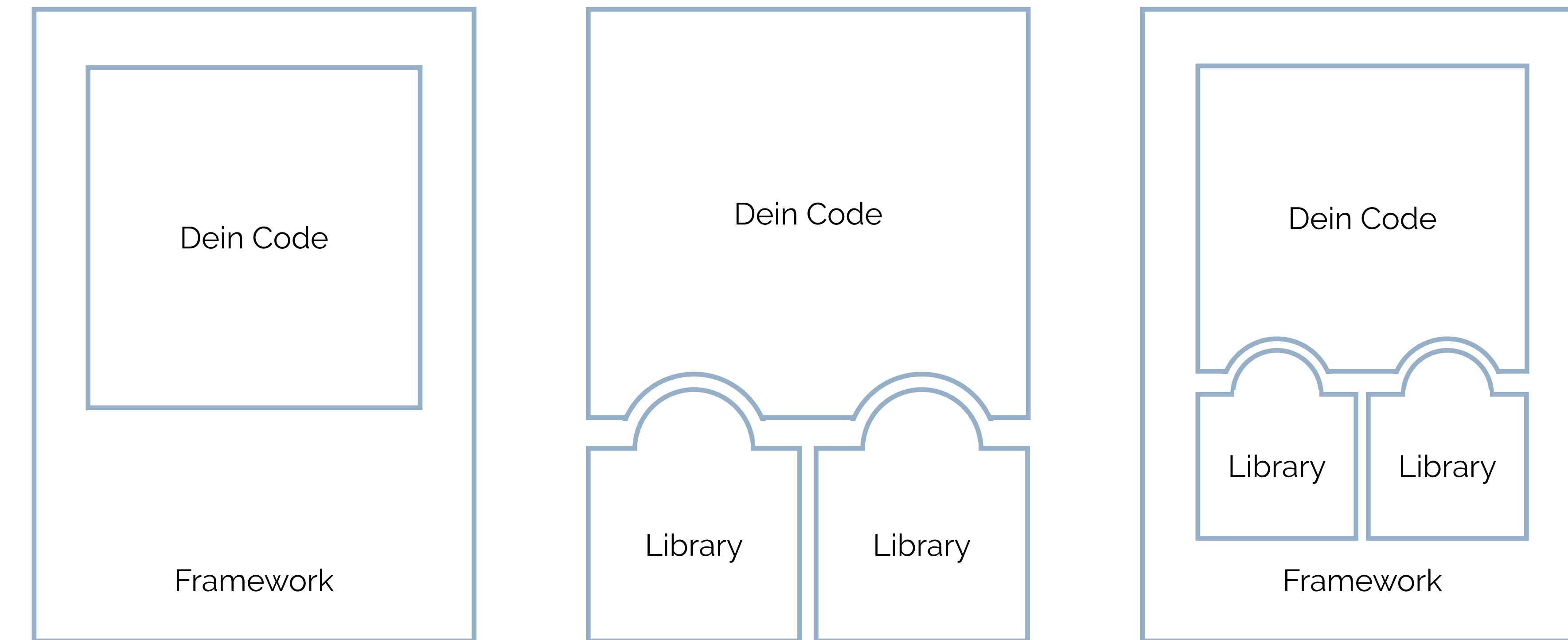
Framework vs. Library

Definition

Boilerplates

Frameworks/
Libraries

Styleguides



Denkprozess

Auswahlverfahren

Definition

Boilerplates

Frameworks/
Libraries

Styleguides

- Wie / mit was wollen wir eine Applikation erstellen?

👍 **TypeScript** ➔ **React** ➔ **Architektur...** (bildet unser framework)

- Welche Funktionen benötigen wir?

👍 **Router** ➔ **React DOM** ➔ **Lodash...** (sind unsere core libraries)

- Wie wollen wir diese darstellen?

👍 **SCSS** ➔ **Ant Design** (sind unsere UI libraries)

- Mit diesen Informationen können wir uns auf ein **Boilerplate** einigen:

```
npx create-react-app my-app
```

Definition
Boilerplates
Frameworks/
Libraries
Styleguides

Online Styleguides

Erfolgsbeispiele

- Geben Brand Guidelines vor, die intern oder extern genutzt werden...
- ...wie Typographie, CSS frameworks, Komponenten, Icons etc.
- Einige sind Open-Source oder öffentlich zugänglich
- Oft wenn wir nach Libraries suchen, finden wir diese in Styleguides, z.B.: Date Picker (Material Design / Lonely Planet)

- Wir werden ein Beispiel mittels Storybook aufbauen

Definition
Boilerplates
Frameworks/
Libraries
Styleguides

GEL

Global Experience Language

Home Guidelines Articles About UX&D



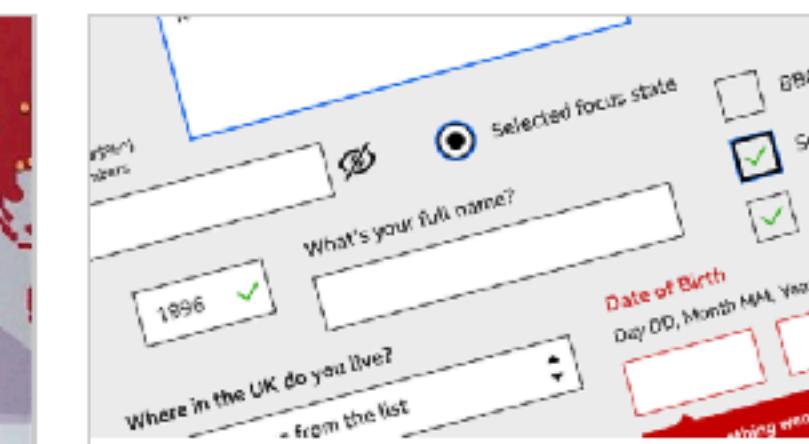
BBC World Service: Designing for Global Users

How does the World Service go about designing and delivering to 40 million users a week in 40 languages?



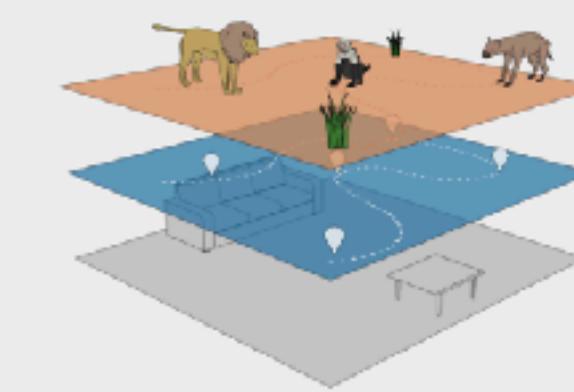
Thomas Anderson
UX Designer

Articles



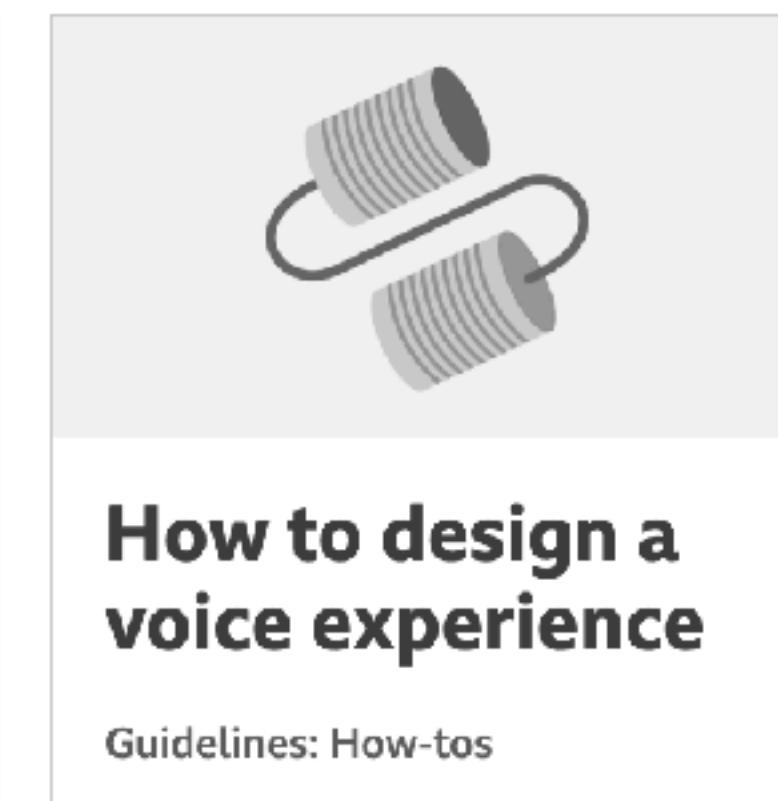
Inputs

Guidelines: Design Patterns



An emergent AR design framework

Articles



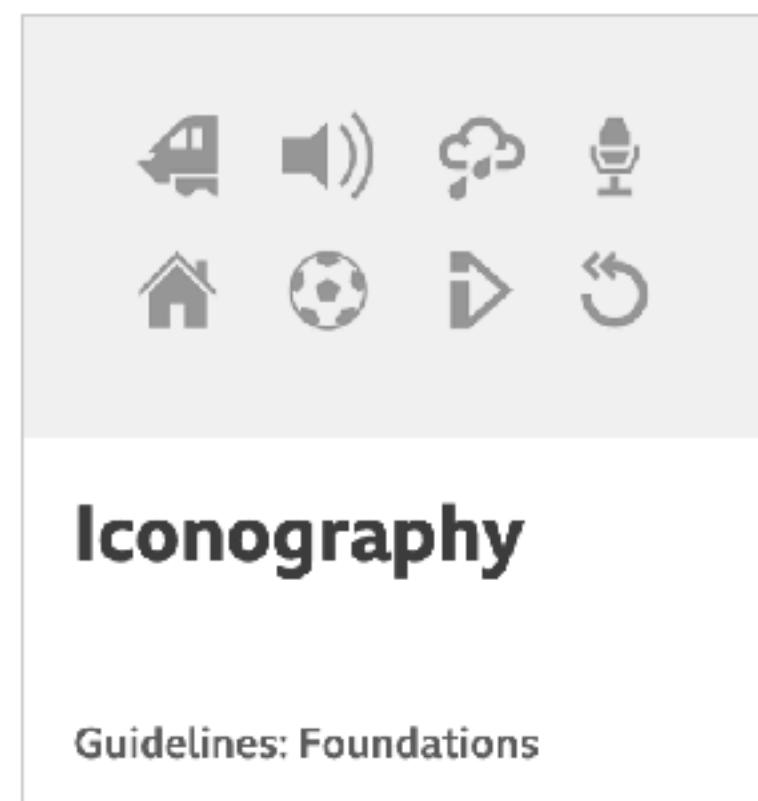
How to design a voice experience

Guidelines: How-tos



User Research for the next design frontier

Articles



Iconography

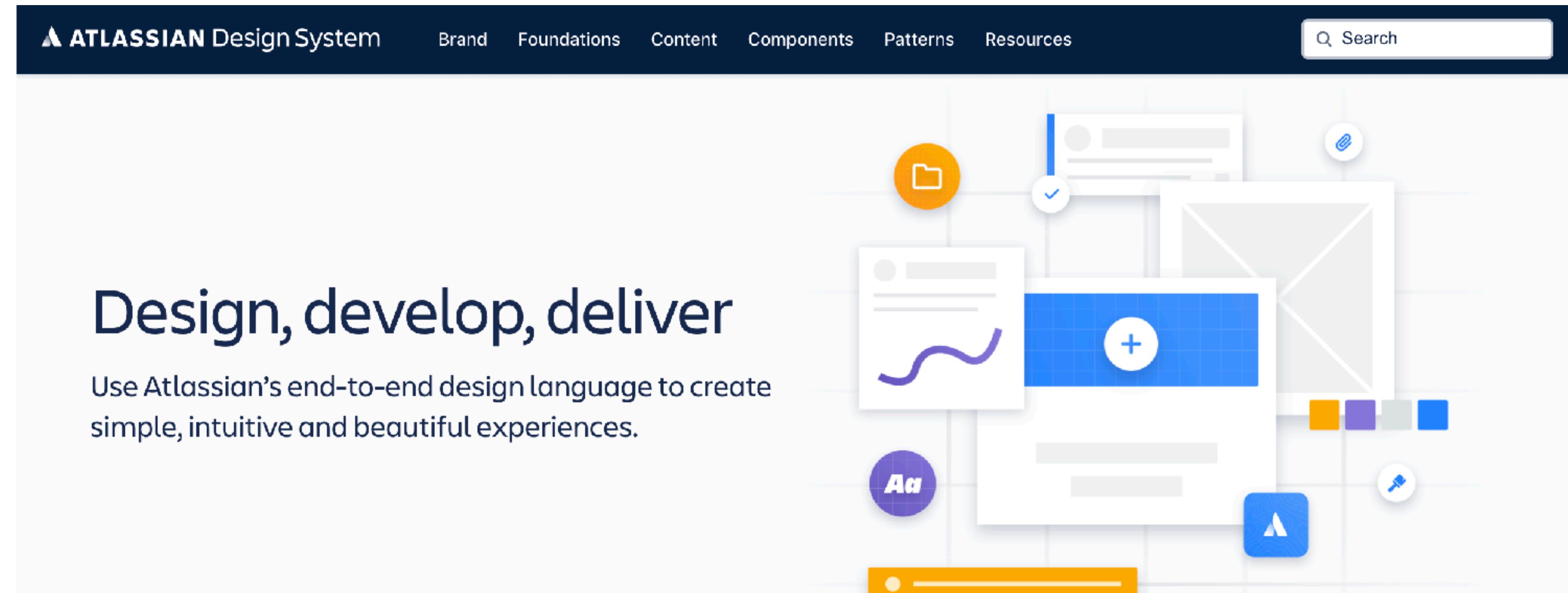
Guidelines: Foundations



What is GEL?

Articles

Definition
Boilerplates
Frameworks/
Libraries
Styleguides



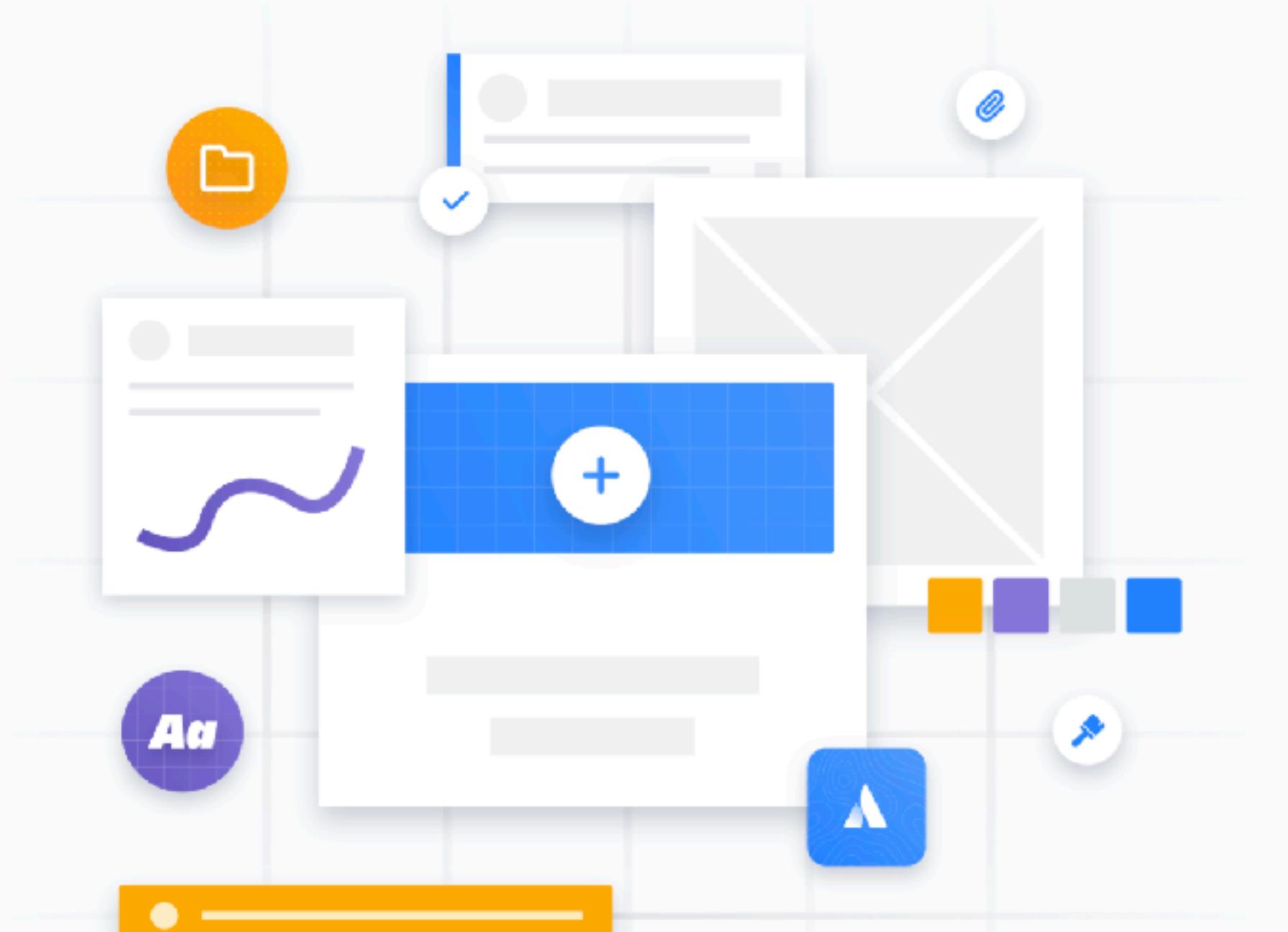
ATLASSIAN Design System

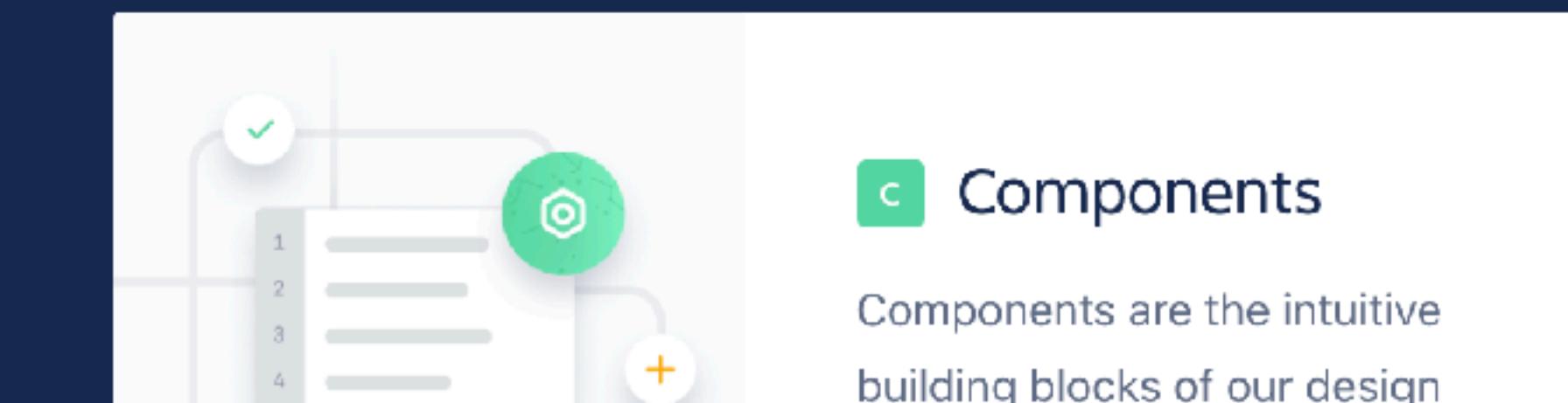
Brand Foundations Content Components Patterns Resources

Search

Design, develop, deliver

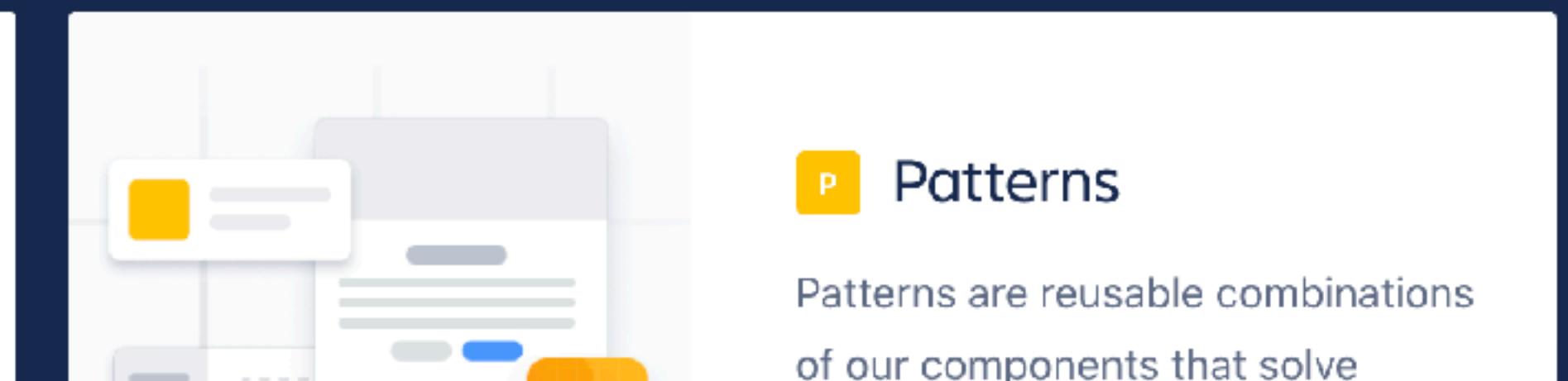
Use Atlassian's end-to-end design language to create simple, intuitive and beautiful experiences.





c Components

Components are the intuitive building blocks of our design



p Patterns

Patterns are reusable combinations of our components that solve

<https://atlassian.design/>

Definition
Boilerplates
Frameworks/
Libraries
Styleguides

The screenshot shows the Primer website homepage. At the top left is the GitHub logo with the word "Primer". At the top right are navigation links: "What's new", "Design ▾", "Development ▾", and "About". The main title "Primer" is in a large, bold, blue font. Below it is a subtitle: "Design, build, and create with GitHub's design system". A paragraph explains that Primer was created for GitHub by GitHub and is now open-sourced. At the bottom are links for "About", "Open-source", and "Community". To the right of the text is a large, stylized blue graphic of a hand holding a paintbrush, with various geometric shapes and icons (checkmark, file, etc.) around it.

<https://primer.style/>

Definition

Boilerplates

Frameworks/
Libraries

Styleguides

Storybook

Lebende Styleguides

- Komponenten werden entkoppelt und stehen im Vordergrund
- Isolierte Entwicklung zeigt dabei schnell Probleme auf
- Mocking und Test-Integration wird von der Community verlangt
- Storybook als Entwicklungsumgebung, wird so nicht überholt
- Dokumentation kann einfach integriert werden
- Entwicklungstools für Komponenten und Hot-reload
- Atomic Design und dann Storybook!

Atomic Design

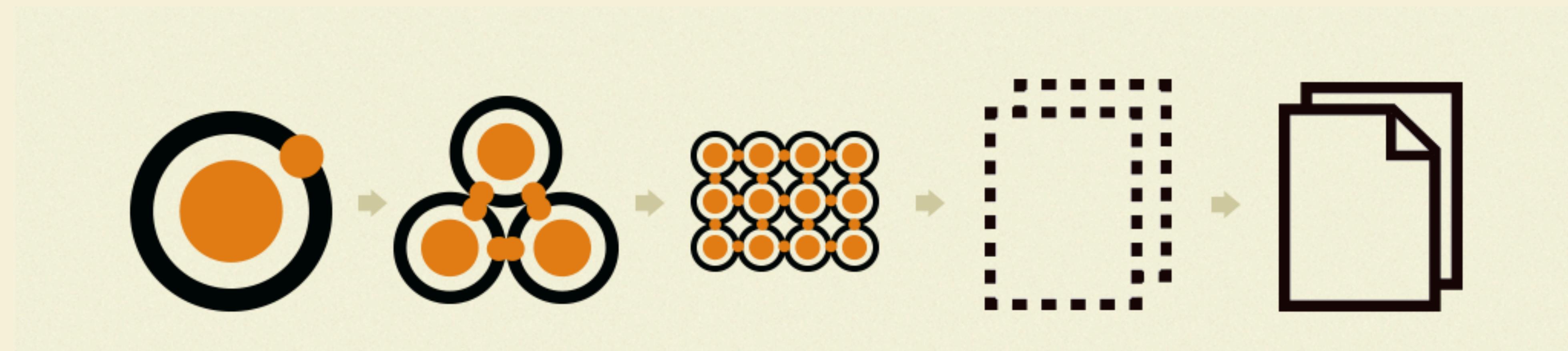


Brad Frost,
Web designer, speaker,
consultant and writer

Atomic Table

html																	output		
base	header																strong		
head	h1-h6																area		
style	hgroup	dt	ol	bdi	data	mark	ruby	sub	sup	map	canvas	col	caption	th	input	progress	param		
title	nav	figcaption	p	bdo	dfn	q	s	time	track	noscript	table	thead	keygen	select	label	details	caption		
body	section	figure	pre	br	em	rp	samp	u	video	script	tbody	tr	button	legend	dialog	col			
address	dd	hr	ul	cite	i	rt	small	var	embed	del	td	thead	keygen	select	label	details	colgroup		
article	div	li	abbr	code	kbd	rtc	span	wbr	object	ins	tfoot	tr	button	legend	dialog	thead	tbody		
footer	dl	main	b																

Übersicht



Kategorisierung

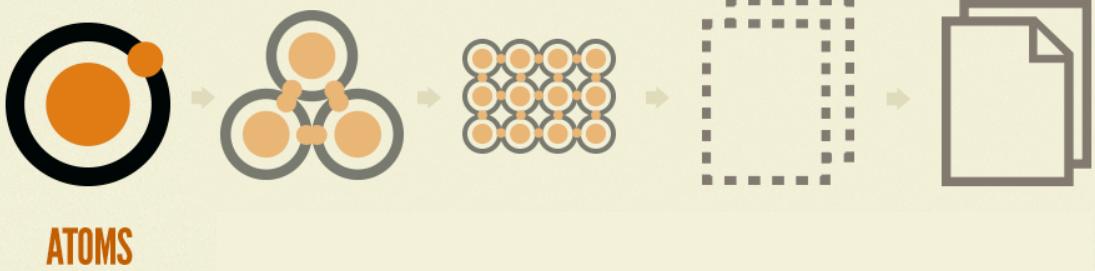
Atome

- Grundbausteine einer Benutzeroberfläche
- Sind unsere HTML tags
- Oder auch abstrakte Elemente wie
Fonts, Farben, Animationen...
- Können nicht kleiner sein
- Nicht sehr hilfreich alleine
- Gut als Referenz

SEARCH THE SITE

ENTER KEYWORD

SEARCH



LABEL

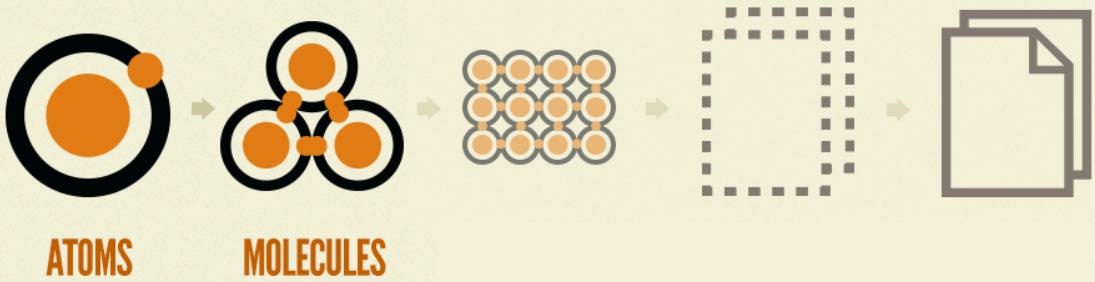
INPUT

BUTTON

Kategorisierung

Moleküle

- Gruppe von Atomen zusammengebunden
- Kleinstes Fundament für eine Komponente
- Konkreter als Atome
- "Tu eins und das gut" Mentalität
- Dient als Rückgrat weiterer Komponenten



SEARCH THE SITE

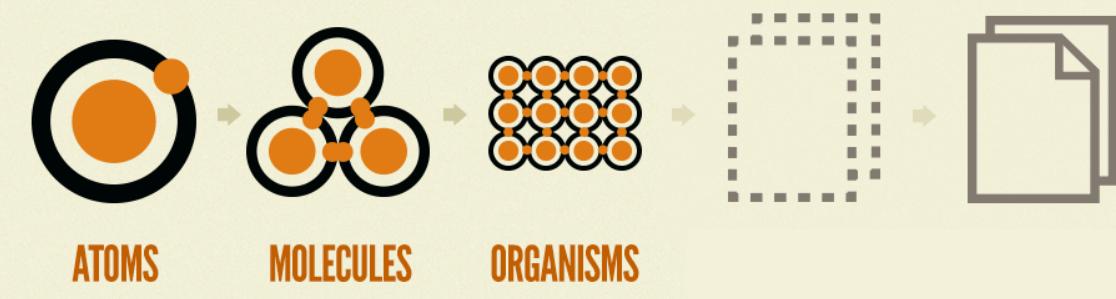
ENTER KEYWORD

SEARCH

Kategorisierung

Organismen

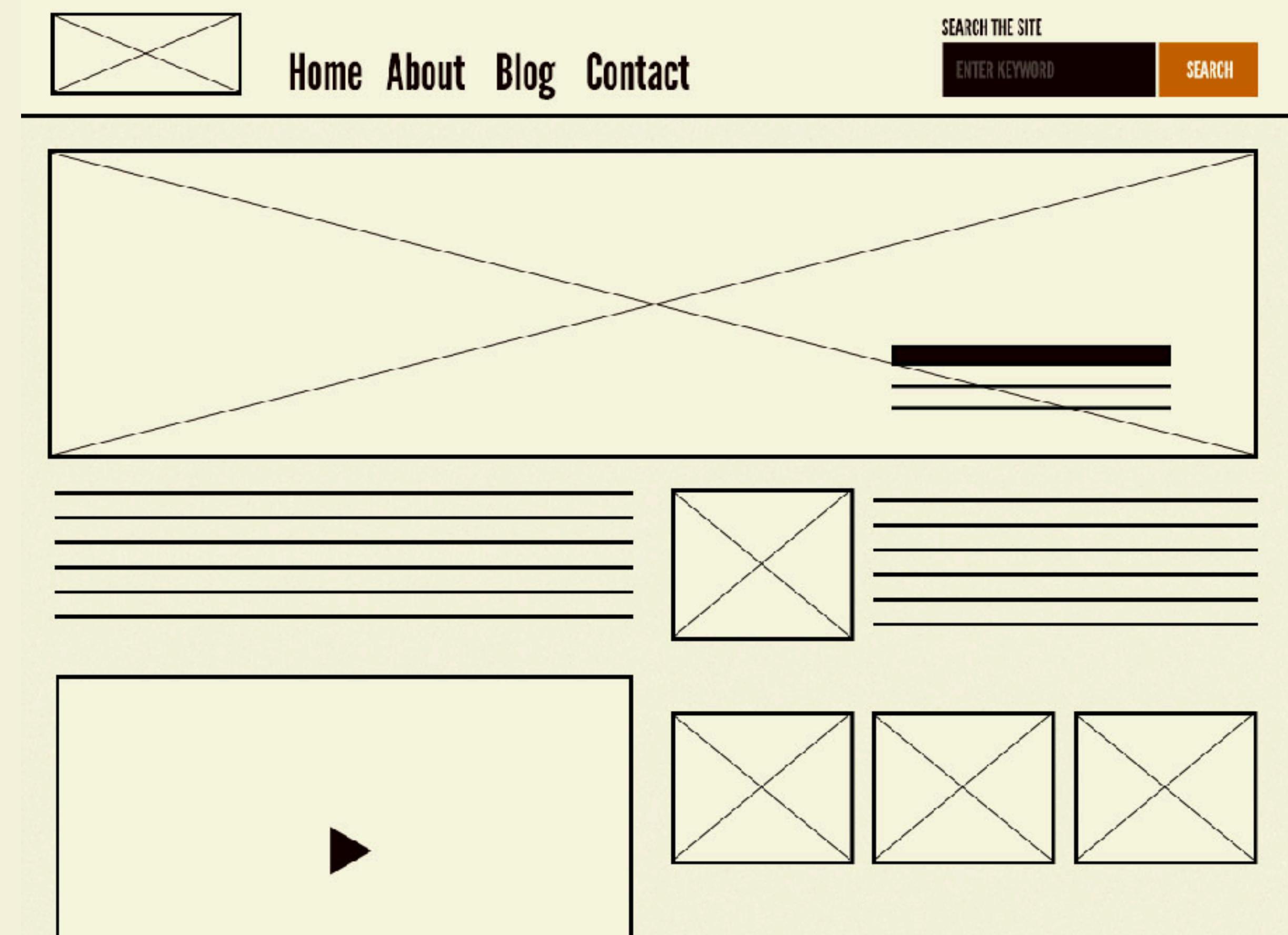
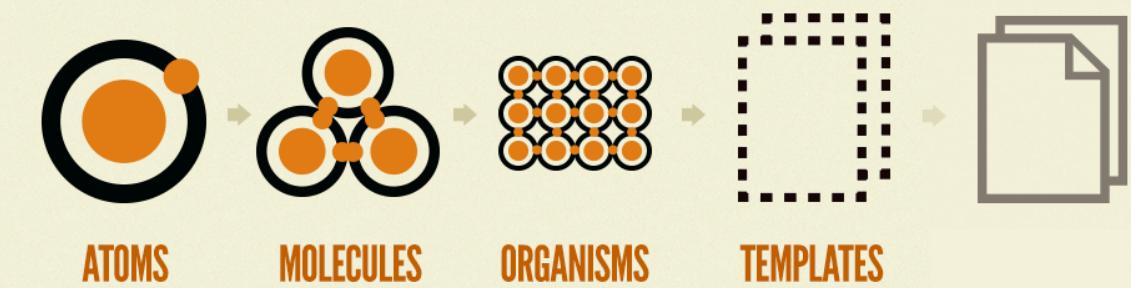
- Gruppe von Molekülen zusammengesetzt
- Es entstehen komplexe Komponenten
- Kann aus den gleichen oder unterschiedlichen Molekülen bestehen
- Unterstützen einzelne, portable und wiederverwendbare Komponenten
- Anordnung kann variieren



Kategorisierung

Templates

- Wir erstellen ein konkretes Beispiel aus Komponenten auf einer Seite zusammen
- Bestehen aus verschiedene Gruppen von Organismen, Molekülen oder einzelnen Atome
- Wireframes, HTML oder Skizzen dienen als Grundlage
- Der Kunde kann Strukturen / Sinn erkennen
- Das Resultat führt zum Produktions-Code

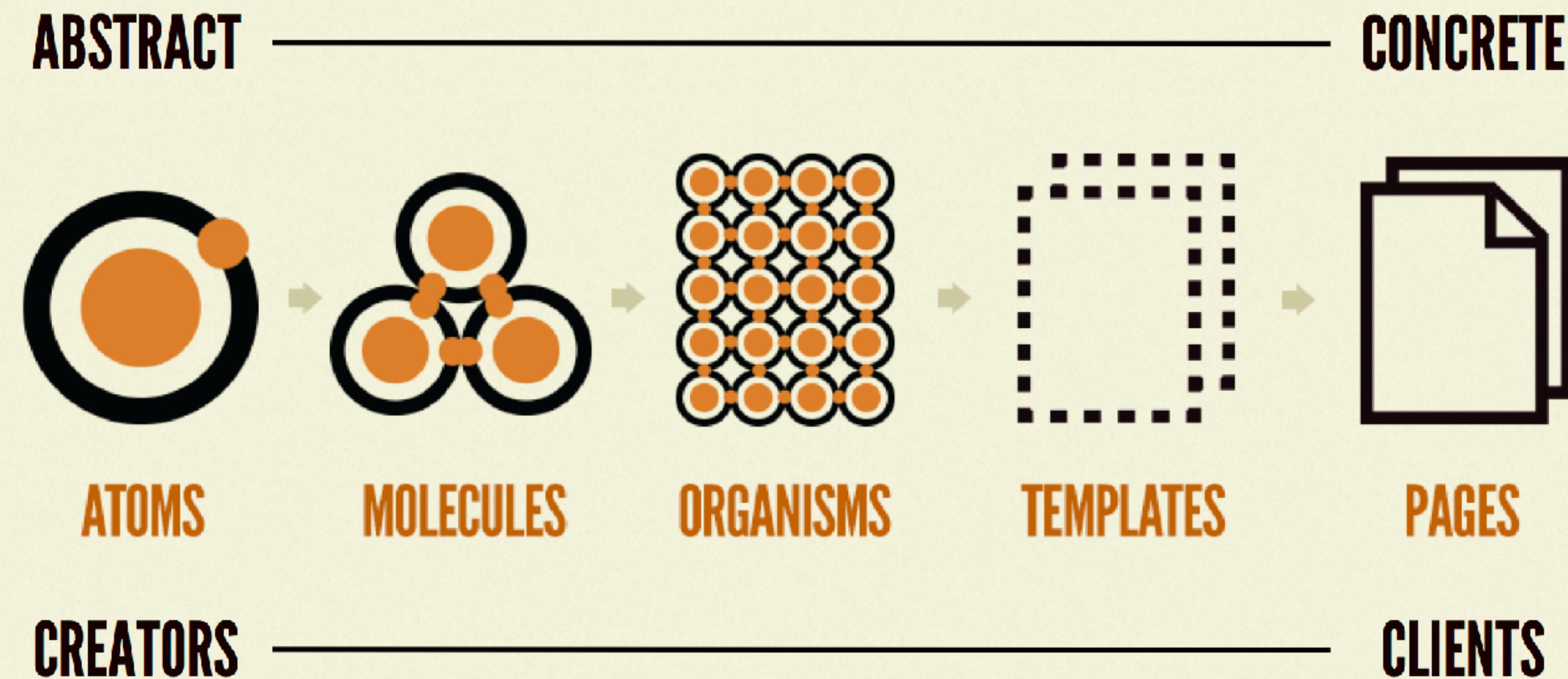


Kategorisierung

Seiten

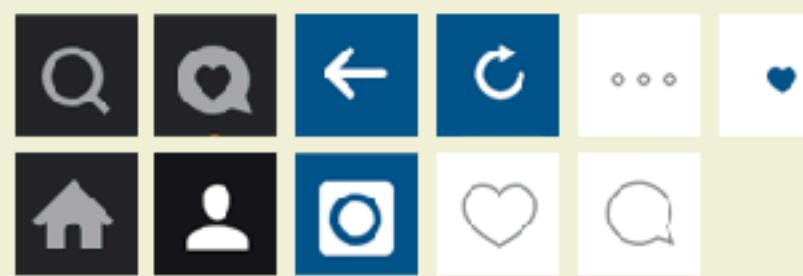
- Wir erstellen eine spezifische Instanz, in der wir Beispieltext mit echtem Inhalt ersetzen
- Inhalt widerspiegelt die Realität
- Wird mit verschiedenen Parteien geteilt
- Testbereites System
- Testet Variationen für Inhalt, Design, Dynamische Elemente

Zusammenfassung





ATOMS



PHOTO

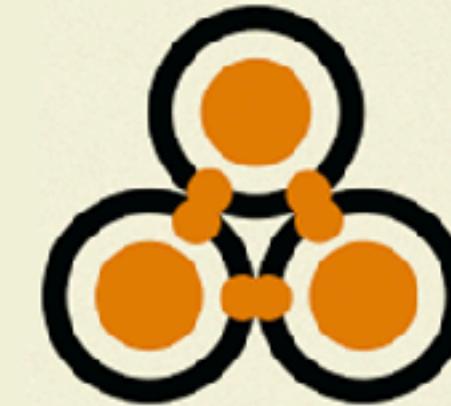
XXXXXX likes

thisistheusersinstagramhandle

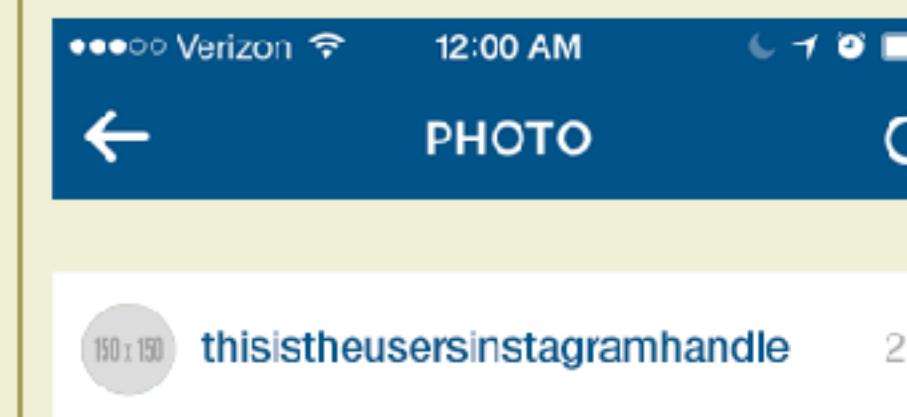
2h

Lorem ipsum dolor sit amet, consectetur
adipisicing

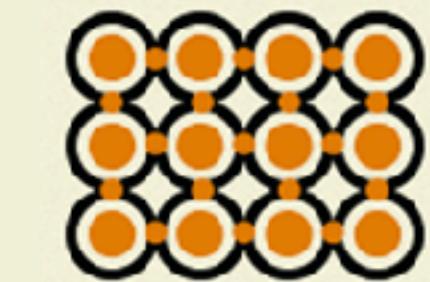
150 x 150



MOLECULES



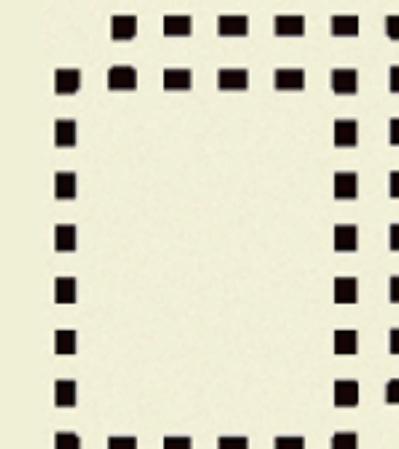
1080 x 1080



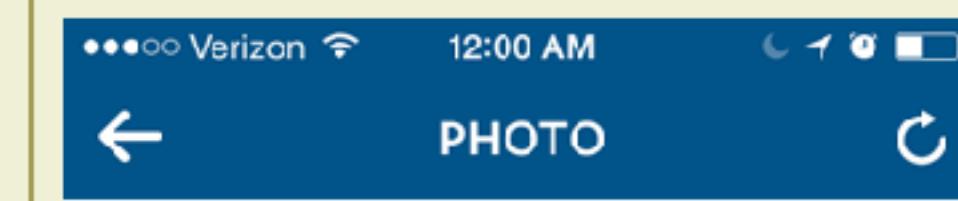
ORGANISMS



1080 x 1080



TEMPLATES



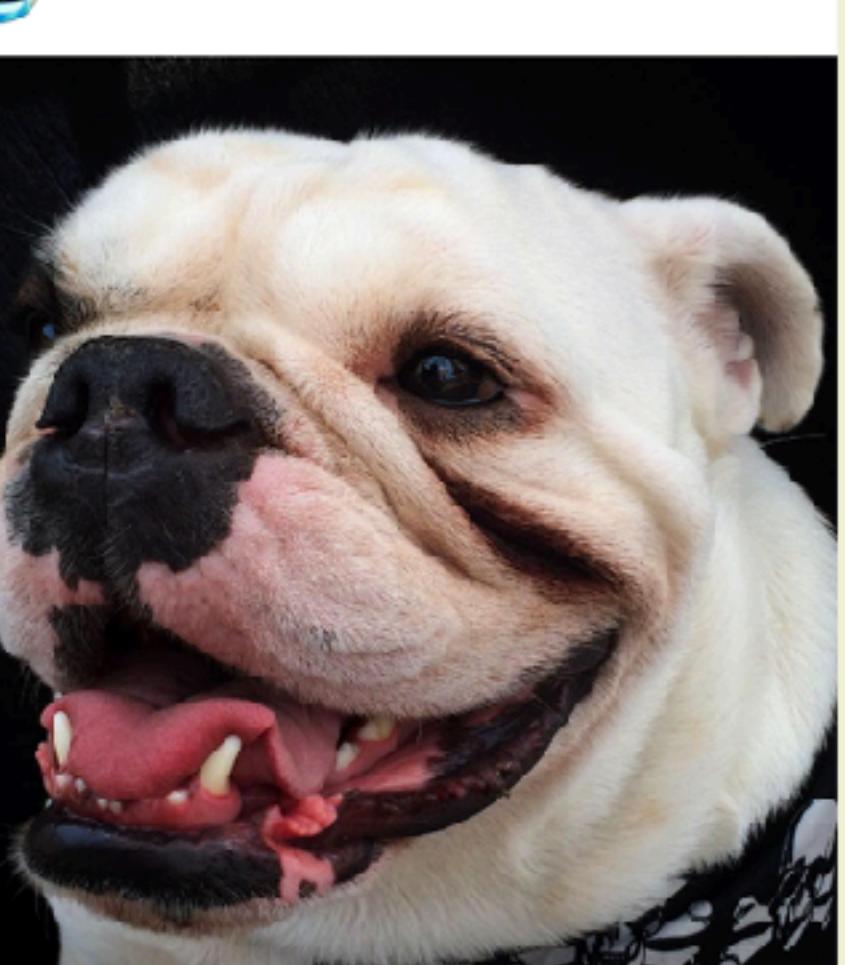
1080 x 1080



PAGES



brad_frost



Atomic Design – Begleitübung

<https://github.com/finalangel/modular-frontend>

- o cd ex3 && npm install && npm run storybook
- o Öffnet: <http://localhost:8001>

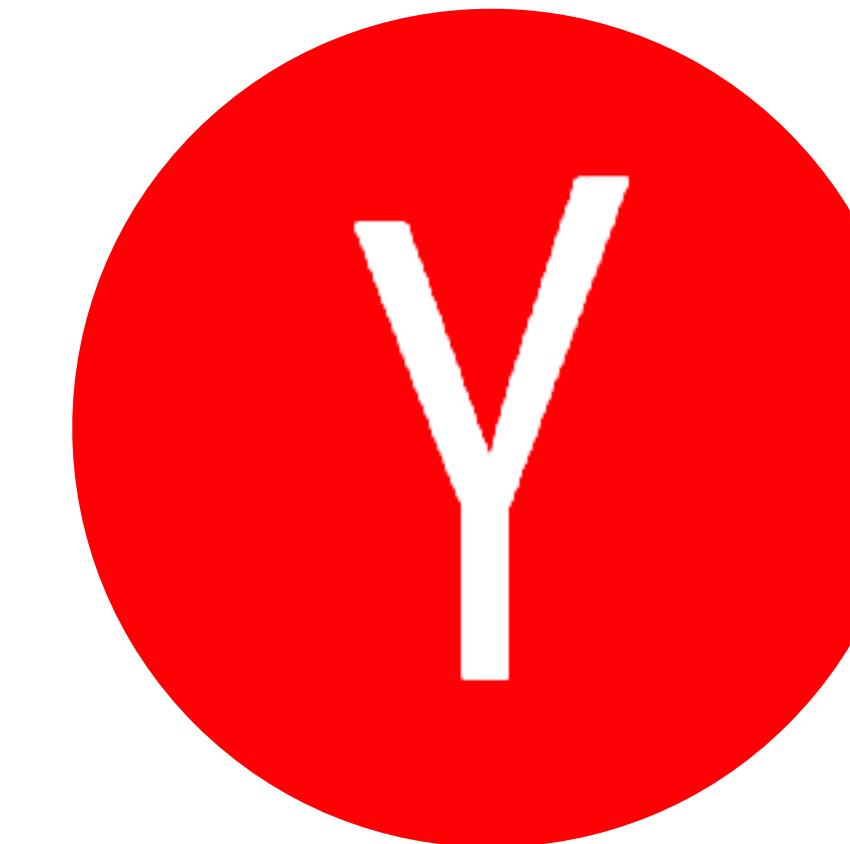
Aufgabe

- o Fügt Stories zu den restlichen Komponenten ein
- o Entscheidet welcher Kategorie sie zugeordnet werden sollen
- o Kann etwas an der Implementierung verbessert werden?
- o Dauer ca. 15-20 Minuten

The screenshot shows a modular frontend application interface. At the top, there is a navigation bar with links: Home, Event, Accessoires, Apparel, and Feautred Items. Below the navigation is a large, bold title "A Big Title". Underneath the title is a placeholder text area with the dimensions "1024 × 300". Below this is a section titled "Intro Copy Headline" with a paragraph of placeholder text: "Fusce vel dui. In enim justo, rhoncus ut, imperdiet a, venenatis vitae, justo. Pellentesque commodo eros a enim. Fusce a quam. Etiam vitae tortor." Below this section are four item cards, each consisting of a gray placeholder image (300 × 300) and a card body. The card bodies have the following structure: "Item Title" (in blue), "Item Category", and the price "\$88.00".

Item Title	Item Category	\$88.00
Item Title	Item Category	\$88.00
Item Title	Item Category	\$88.00
Item Title	Item Category	\$88.00

BEM – Block; Element; Modifier



Yandex,
A technology company,
from Russia with ❤️

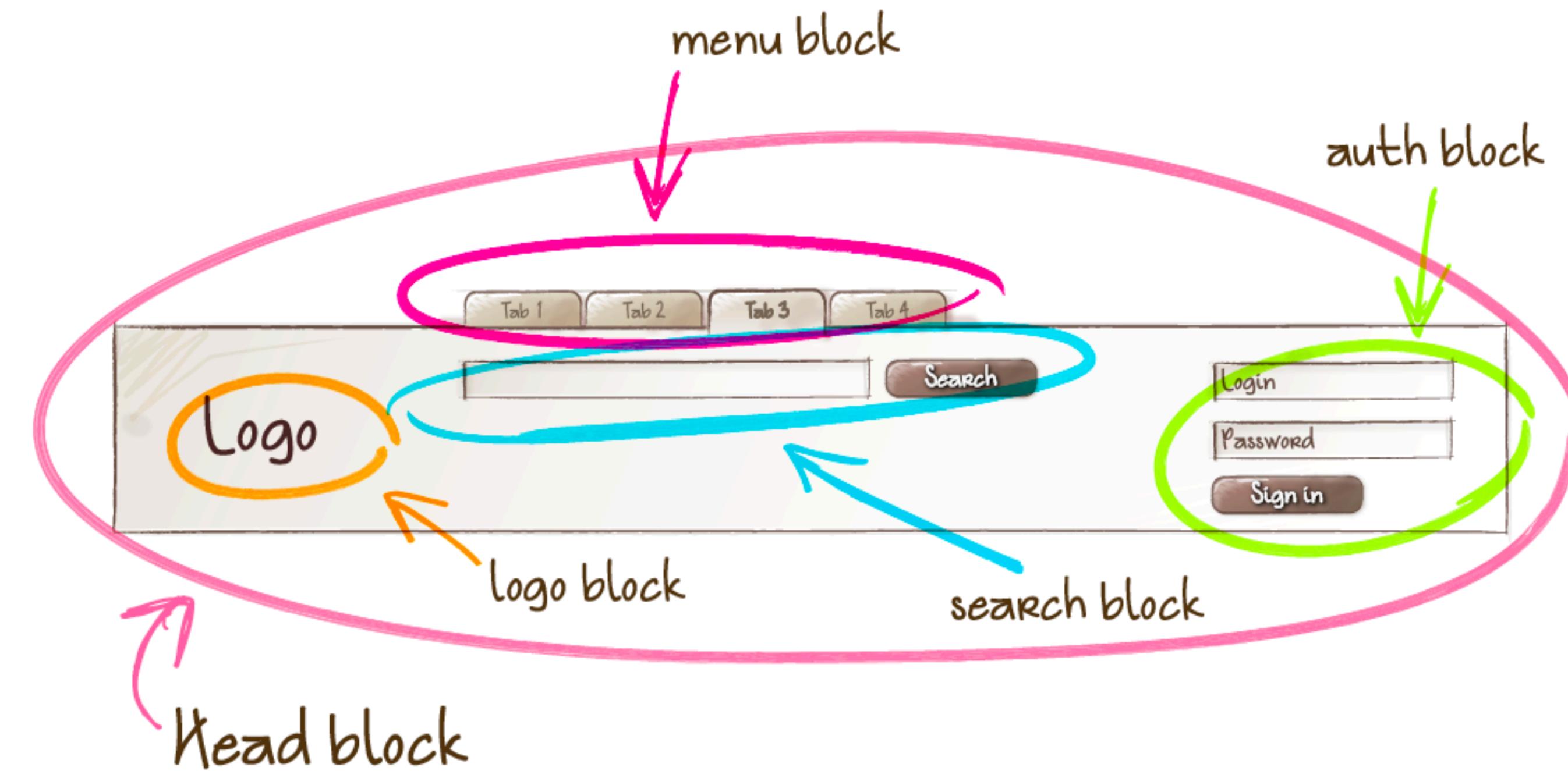
Definition

Block

- Unabhängige Komponente die wiederverwendet werden kann
- Beschreibt seine Existenzberechtigung ➔ "Was ist es?"
- Beschreibt nicht wie es aussieht ➔ "red, big"
- Ein Block sollte keine Geometrie definieren ➔ "padding, margin"
- Verschachteln ist erlaubt ➔ "header" ➔ "form" ➔ "login"

Beispiel

Block



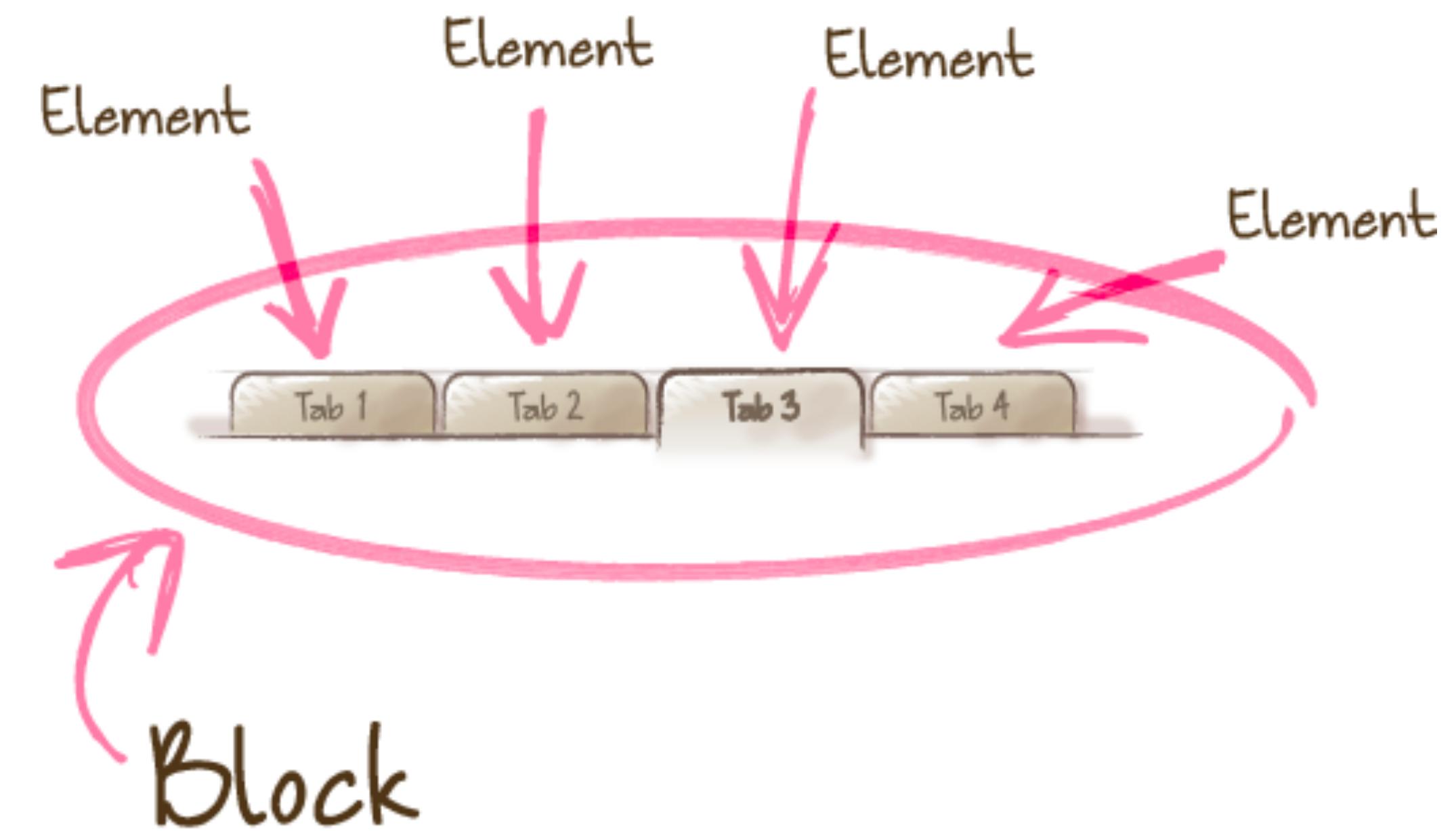
Definition

Element

- Beschreibt seine Existenzberechtigung ➡ "Was ist das?"
- Beschreibt nicht wie es aussieht ➡ "red, big"
- Beschreibt nicht welchen Zustand es hat ➡ "active, disabled"
- Block und Element müssten getrennt ersichtlich sein ➡ "block_element"
- Verschachteln ist erlaubt ➡ "search-form" ➡ "search-form_input" ➡ "search-form_button"
- Ist **immer** ein Teil eines Blocks

Beispiel

Element



Wann Block, wann Element

Worin liegt der Unterschied?

Block

- Wenn ein Codeabschnitt wiederverwendet werden kann und dieser nicht von anderen Komponenten abhängig ist.

Element

- Wenn ein Codeabschnitt nicht separat verwendet werden kann und auf andere Komponenten angewiesen ist. (Benötigt einen Block als Überelement)

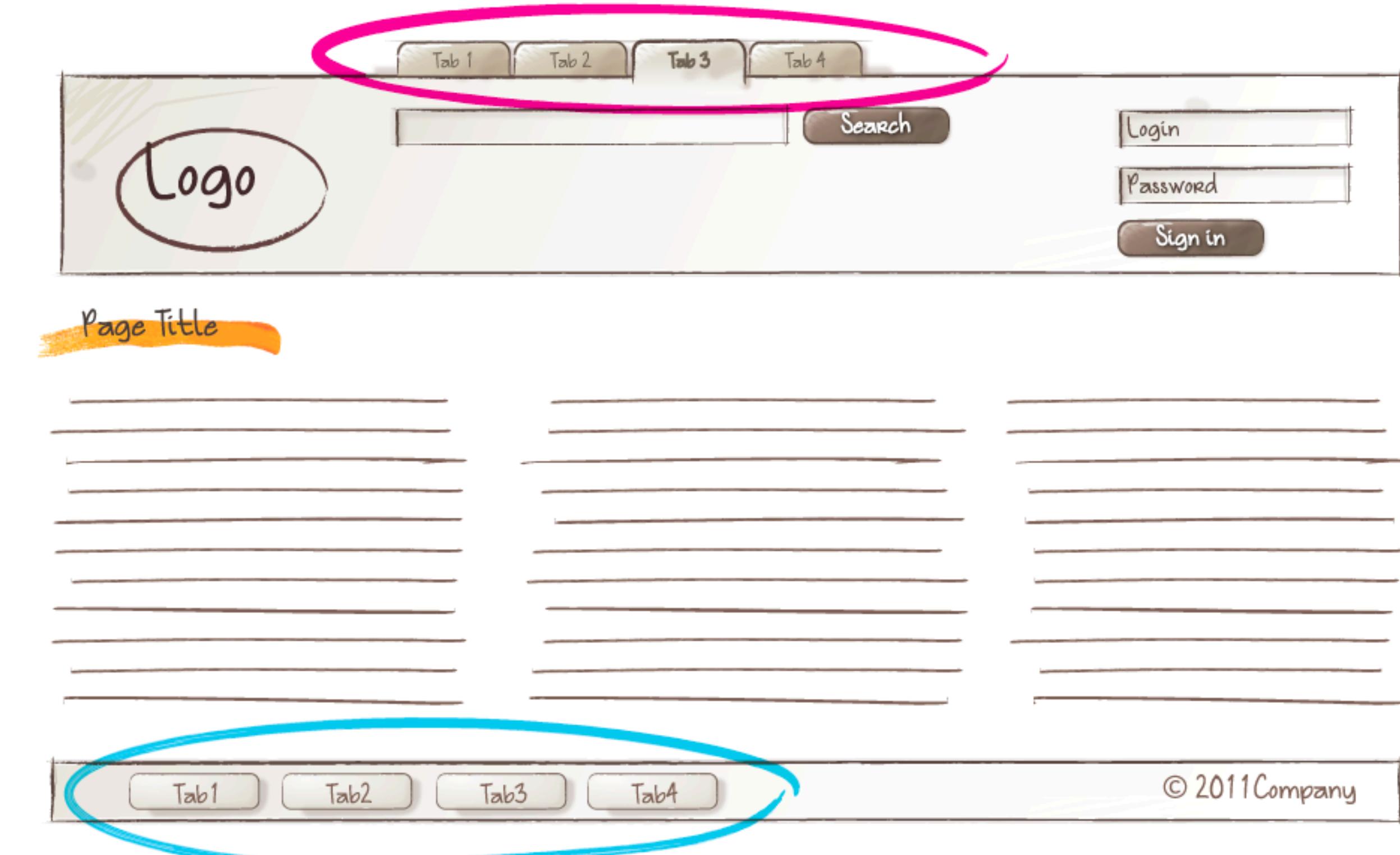
Definition

Modifier

- Hier definieren wir, wie er aussehen soll oder welches Verhalten angenommen werden soll: welche Grösse, welches Theme, Unterschiede zu anderen Elementen etc.
- Der Modifier wird mit einem einfachen "_" gekennzeichnet:
`block-name_element-name_modifier_name`
- Kann auf Block oder Element angewendet werden:
`box--active, box_header--sticky`

Beispiel

Modifier



Vermischung

Mix

- Block und Elemente können kombiniert werden damit ein neues UI Element entsteht, somit kann der Stil einfach getrennt werden:

```
<div class="header">
  <form class="form header__form">
    <input type="search" class="input header__input" />
  </form>
</div>
```

Benennung

block-name__element-name_modifier-name

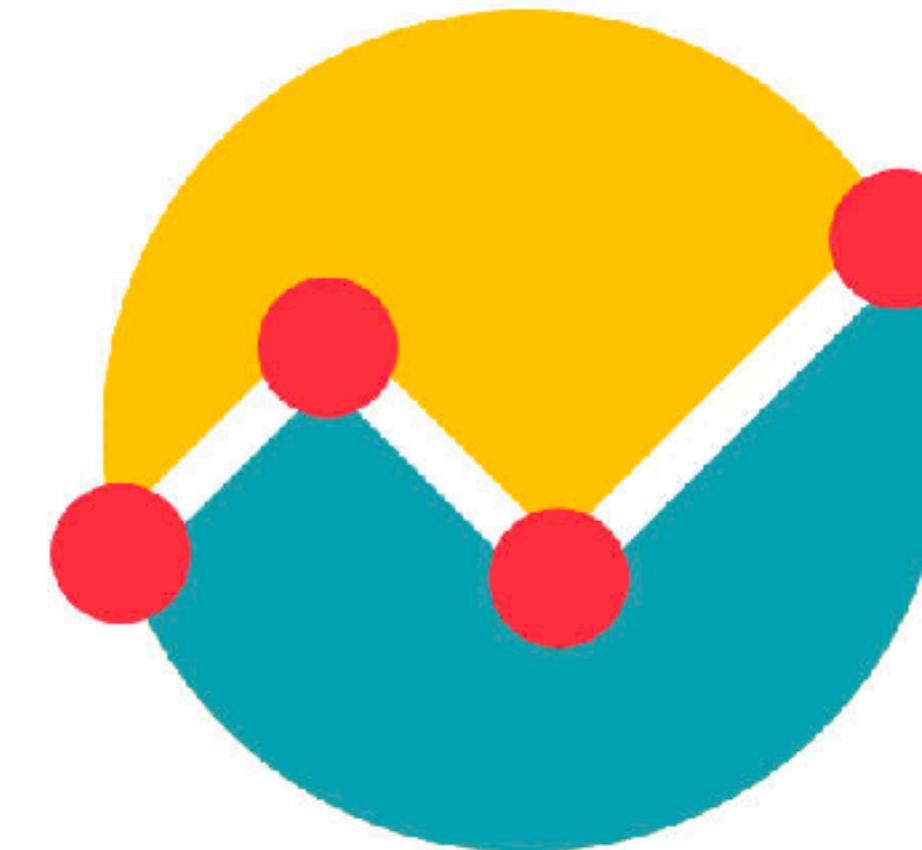
- Gängige Regeln wie bei Variablen (keine Sonderzeichen, nicht mit Zahl beginnen...)
- Alles in Kleinschreibung und in englischer Sprache
- Wort Trennung mit einem Bindestrich “-”
- “Element” Trennung mit zwei Unterstrichen “__”
- “Modifier” Trennung mit einem Unterstrich “_”

Zusammenfassung

Block / Element / Modifier

- Block als **Grundbaustein**
- Element als **Struktur** eines Blocks
- Modifier als **Zustand** eines Blocks oder Elementes
- Benennung: block-name_element-name_modifier-name

Alternativen



ATOMIC
OOBEMITSG

Verschmelzung verschiedener
Methodologien

[Website](#)

ITSG

Grundsatz

[ATOMIC OOCSS BEM ITSG]



Wie funktioniert CSS?

Challenge

- Block und Elemente können kombiniert werden damit ein neues UI Element entsteht, somit kann der Stil einfach getrennt werden:

```
> cat base.css
.red {
    color: red;
}

.blue {
    color: blue;
}

> cat index.html
<div class="red blue">Text</div>
<div class="blue red">Text</div>
```

Welche Farbe hat der "Text"?

- A) Erste rot, zweite blau
- B) Erste blau, zweite rot
- C) Beide blau
- D) Beide rot

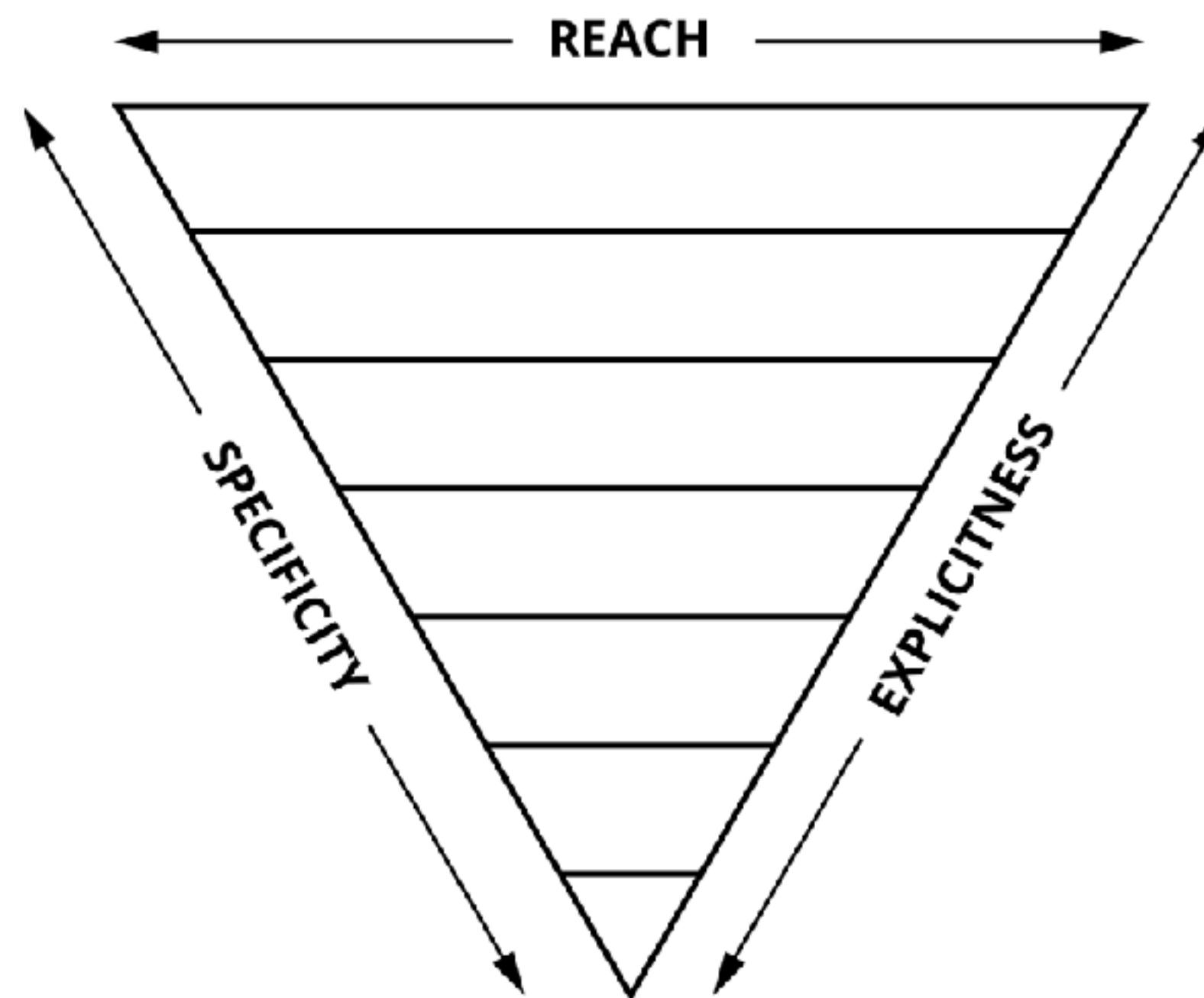
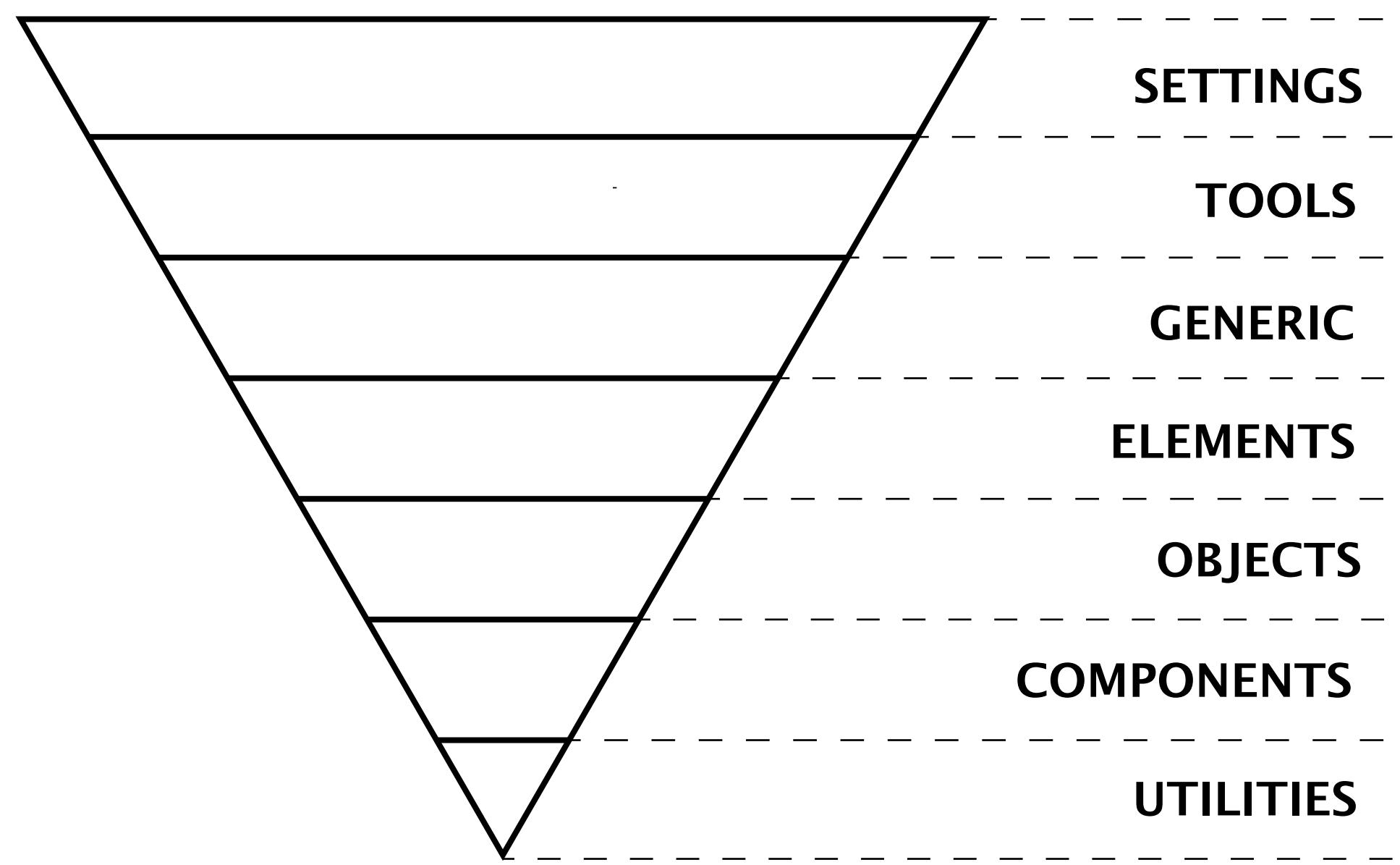
So was ist ITSG?

Definition

Inverted Triangle & Specificity Graph

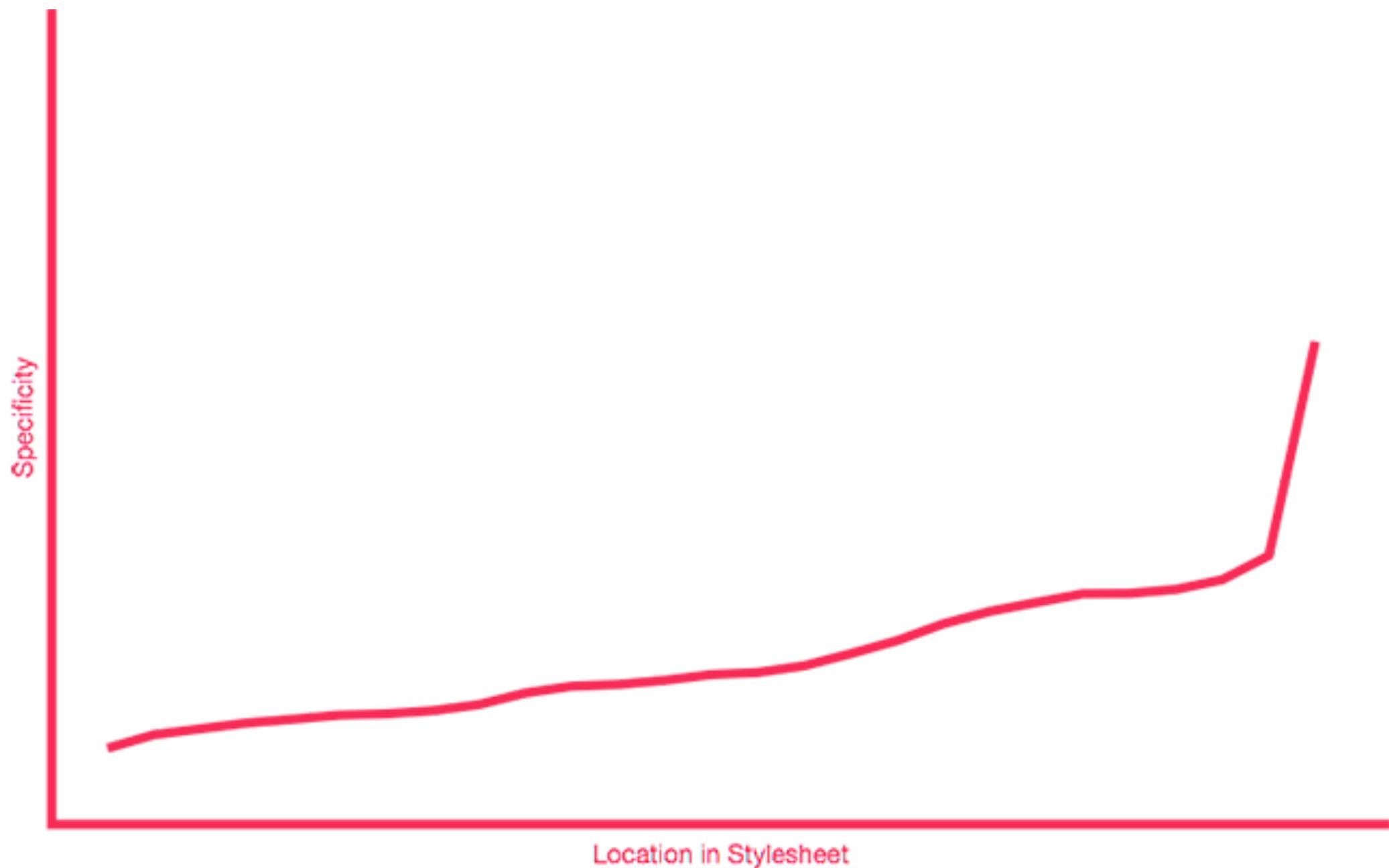
Inverted Triangle

Specificity von Oben nach Unten



Specificity Graph

Wie komplex ist euer CSS



Abschluss – Begleitübung

<https://github.com/finalangel/modular-frontend>

- Wir bleiben bei ex3!

Aufgabe

- Passt die Komponenten an denn BEM Stil an
- Verbessert die Komponenten weiter damit der resultierende Specificity Graph flach wird
- Nehmt weitere Verbesserungen vor
- Dauer ca. 15-20 Minuten

The screenshot shows a website template with the following structure:

- Header:** A large, dark grey header section containing a placeholder image with dimensions 1024×300 .
- Navigation:** A navigation bar with links: Home, Event, Accessoires, Apparel, and Feautred Items.
- Main Content:** A section titled "Intro Copy Headline" with placeholder text: "Fusce vel dui. In enim justo, rhoncus ut, imperdiet a, venenatis vitae, justo. Pellentesque commodo eros a enim. Fusce a quam. Etiam vitae tortor." Below this are four item cards, each with a placeholder image of size 300×300 and the following details:
 - Item Title:** Item Category \$88.00
 - Item Title:** Item Category \$88.00
 - Item Title:** Item Category \$88.00
 - Item Title:** Item Category \$88.00
- Footer:** A dark footer section with the text "www.adobe.com".

Aus der Produktion

Deep Dive

Beispiele aus der Produktion

- [flavours.dev](#) & Gatsby
- [divio.com](#) & Gatsby
- Control Panel / Control Panel Frontend

DIVIO

PRODUCT SERVICES DEVELOPERS PRICING SIGN IN [SIGN UP FOR FREE](#)

The web platform that **has a team that supports you**

Our easy-to-use platform lets you focus on your apps and takes care of any cloud - even on-premises. It is trusted by leading enterprises to transform and grow their digital presence.

Book a personal demo or start a live chat - we can't wait to meet you.

[Watch a demo](#) [Sign up for free](#)

You'll talk directly to our Divio architects, we'll take all the time you need.

TRUSTED BY

 **Fidelity**
INTERNATIONAL

 **Zürcher
Kantonalbank**

 **LWL**

 **APGISGA**

 **WALKER &
DUNLOP**

 **Prezi**



Vielen Dank

& ein erfolgreicher Abschluss



Appendix A

Stock images

<https://unsplash.com/>

Repository

<https://github.com/finalangel/modular-frontend/>

Einführung

<http://www.lego.com/>

HTML & CSS

<https://csswizardry.com/2013/04/shame-css/>

<https://www.smashingmagazine.com/2011/12/an-introduction-to-object-oriented-css-oocss/>

<https://github.com/stubbornella/oocss/>

<http://smacss.com/>

<https://javascript.info/dom-navigation>

JavaScript

<https://stateofjs.com/>

<https://www.npmjs.com/>

<https://www.smashingmagazine.com/2020/05/styling-components-react/>

Frameworks

<http://saijogeorge.com/brand-style-guide-examples/>

Styleguides

<http://www.bbc.co.uk/gel>

<https://atlassian.design/>

<https://primer.style/>

<https://developer.apple.com/design/human-interface-guidelines/>

<https://storybook.js.org/>

Appendix B

Atomic Design

<https://atomicdesign.bradfrost.com/>

<https://madebymike.github.io/html5-periodic-table/>

BEM

<https://en.bem.info/>

ITSG

<https://css-tricks.com/atomic-oobemitscss/>

<https://jonassebastianohlsson.com/specification-graph/>

<https://csswizardry.com/2014/10/the-specificity-graph/>

<https://www.xfive.co/blog/itcss-scalable-maintainable-css-architecture/>

Beispiele

<https://www.divio.com/>

<https://www.flavours.dev/>

<https://control.divio.com/>

<https://angelo.dini.dev/>